IBM ESS and IBM DB2 UDB Working Together

Layout DB2 UDB on the ESS and maximize performance

Map system resources to database activity on the ESS

Make Copy Services part of your backup strategy

Barry Mellish
John Aschoff
Bryan Cox
Dawn Seymour

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM ESS and IBM DB2 UDB Working Together

October 2001

SG24-6262-00

Take Note! Before using this information and the product it supports, be sure to read the general information in “Special notices” on page 131.

First Edition (October 2001)

This edition applies to DB2 UDB Version 7.2, AIX Version 4.3, and the ESS F Model with Copy Services.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. QXXE Building 80-E2
650 Harry Road
San Jose, California 95120-6099

Contents

Figures

Tables

Preface

This IBM Redbook will give you sufficient information to effectively use the IBM Enterprise Storage Subsystem with DB2 Universal Database (DB2 UDB). Much of the information regarding how to layout your database that is presented here will apply to both UNIX and Windows environments. However, specific examples and commands will be given only for the AIX platform.

Chapter 1 supplies an overview of the products — IBM Enterprise Storage Subsystem, AIX, and DB2 Universal Database — that we used in our environment.

Chapter 2 introduces the terminology that will be used throughout the remainder of the book. We also describe several overriding concepts that need to be considered when setting up an Enterprise Data Management Server using ESS and DB2 UDB.

Chapter 3 further discusses the configuration options available for these products, discussing the advantages and disadvantages of each option in terms of both performance and manageability.

Chapter 4 discusses the guidelines that we recommend for sizing an Enterprise Storage Subsystem for use with DB2 Universal Database.

Chapter 5 discusses the challenges of mapping the system resources to the database activity in an ESS environment. Specific examples include the use of virtual pathing, which introduces an additional layer of resource identification in which existing AIX customers are used to.

Chapter 6 discusses the diagnostic tools which are available and how to use them when working with performance issues.

Chapter 7 discusses backup and recovery and explains some of the new features available with DB2 UDB V7.2, which enables some of the ESS capabilities. General backup and recovery features are also covered here.

Appendix A includes descriptions of the testing that was done as part of this IBM Redbook project. These tests were executed to determine the comparative effect of multiple data placement options as well as to compare the effects of various database tuning parameters on performance.

This book is intended to be read by systems administrators, storage specialists, database administrators and database specialists. We assume a base level of understanding of one of the areas addressed: ESS, AIX and/or DB2, and intend that this book helps you to understand your area of expertise in a wider context.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Barry Mellish is a Project Leader at the International Technical Support Organization, San Jose Center. He has coauthored seven previous redbooks and has taught many classes on storage subsystems. He joined the IBM UK 17 years ago, and before joining the ITSO, he worked as a Senior Storage Specialist on the Disk Expert team in EMEA.

John Aschoff has worked for IBM for longer than he can remember. He has specialized in the field of storage performance, most recently in open systems and database, and has published several white papers and professional journal articles. John is currently the Project Manager for database systems in the Storage Systems Group Open Systems Laboratory in San Jose.

Bryan Cox is the NUMA-Q Teraplex Center Team Lead in Beaverton, Oregon. He has worked for many years years with high performance computers at Sequent. He has been responsible for numerous database benchmarks and performance evaluation studies.

Dawn Seymour is a Senior Data Management Software IT Specialist from Portland, Oregon. She has over 20 years of experience working in the I/T industry and joined IBM 17 years ago as a Systems Engineer, supporting application development products, knowledge based systems, and document imaging products. Her current areas of expertise include DB2 UDB on Intel and Unix platforms, Data Joiner, Data Propagator and DB2 UDB Enterprise-Extended Edition (EEE).

Thanks to the following people for their invaluable contributions to this project:

This project would not have been possible without the help of many people. In particular the project team wants to thank **Kwai Wong** of IBM Toronto for her invaluable help in running the benchmarks and **Dale McInnis** for his contribution to the Database backup and recovery chapter.

From IBM San Jose we want to thank:

- Nicki Rich, Storage Systems User Centered Design
- Neena Cherian, I/O Subsystem Performance
- Paul K. Lee, Manager Database and Application Verification
- David (Yang Seon) Kim, Performance Evaluation

From the IBM DB2 Toronto Lab we want to thank:

- Berni Schiefer, Manager DB2 Performance and Advanced Technology
- Karen Sullivan, DB2 UDB Performance
- Haider Rizvi, DB2 UDB Performance
- Anoop Sood, Toronto Development Support Services
- Kwai Wong, DB2 UDB Performance
- Michel Boisvert, Toronto Development Services
- Dale McInnis, Manager DB2 Utilities Development
- Dwaine Snow, DB2 Lab Services, Toronto

From the IBM marketing support we want to thank:

- Randy Holmes, Team lead, DB2 UDB Customer Benchmarking team
- Gus Branish, DB2 UDB Customer Benchmarking team
- Claude Pelletier, pSeries Product Specialist, Beaverton, Oregon
- Jack Baker, Solutions Development Engineer, IBM Beaverton

- Sean Byrd, Data Management Specialist, Salt Lake City

Special notice

This publication is intended to help systems administrators, storage specialists, database administrators and database specialists to layout an IBM DB2 UDB database on the IBM ESS Storage Subsystem. The information in this publication has not been subject to formal verification and is provided AS IS. See the PUBLICATIONS section of the IBM Programming Announcement for DB2 UDB Version 7.2, AIX Version 4.3, and the ESS F Model with Copy Services for more information about what publications are considered to be product documentation.

IBM trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

e (logo)®
IBM ®
AIX®
AIX 5L™
Balance®
BookMaster®
Cross-Site®
CUA®
DB2®
DB2 Connect™
DB2 Universal Database™
DFS™
DFSMS/MVS®
DFSMSdss™
Enterprise Storage Server™
Enterprise Systems Connection Architecture®
ESCON®
Everyplace™
FICON™
FlashCopy™
IMS™
Informix™
Manage. Anything. Anywhere®
MVS™
MVS/DFP™
MVS/ESA™
Redbooks
Redbooks Logo
NetView®
NUMA-Q®
OS/2®
OS/390®
OS/400®
Perform™
Planet Tivoli®
pSeries™
RACF®
RAMAC®
Redbooks™
RS/6000®
S/390®
Seascape®
Sequent®
SP™
StorWatch™
System/370™
System/390®
Tivoli®
Tivoli Enterprise™
TME®
TotalStorage™
Working Together®

Comments welcome

Your comments are important to us!

We want our IBM Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  `ibm.com/redbooks`

- Send your comments in an Internet note to:

  `redbook@us.ibm.com`

- Mail your comments to the address on page ii.

1

Introduction to the ESS, DB2 UDB and AIX

Complex systems are by nature made up of multiple parts. In order for these systems to be successful, the parts need to work together in a coordinated fashion, each taking advantage of the other's strengths. In this part of the book we give an overview of three major parts of an Enterprise Database Server: the storage subsystem (Enterprise Storage Subsystem or ESS), the operating system (AIX) and the database management system (DB2 Universal Database). We discuss the benefits that customers are realizing, using this combination of technology, including manageability and the ability to handle a wide-varieties of workloads.

1.1 Introduction to the IBM Enterprise Storage System

The first of IBM’s Enterprise Storage Systems (ESS) were introduced in 1999 to meet the need for high performance, scalable, flexible, and available storage systems with advanced management capabilities. The most recent product in IBM’s Seascape architecture family, the ESS, sometimes referred to as Shark, is a SAN-ready disk storage system providing universal access across all major server platforms. It employs advanced hardware and software technologies to deliver breakthrough levels of performance and maximize data sharing across the enterprise.

An enhancement to the ESS, ESS Copy Services, provides three varieties of replication of mission critical data. FlashCopy delivers near-instantaneous, non disruptive, point-in-time copies within the ESS; Peer-to-Peer Remote Copy (PPRC) implements dynamic synchronous mirroring to a remote ESS; and in IBM z-series environments, asynchronous copying to a remote site is possible using Extended Remote Copy (XRC).

1.1.1 The Seascape architecture

Seascape is a blueprint for comprehensive storage solutions optimized for a connected world. The Seascape architecture integrates leading technologies from IBM — including disk storage, tape storage, optical storage, powerful processors, and rich software function — to provide highly reliable, scalable, versatile, application-based storage solutions that span the range of servers from PCs to supercomputers.

At its heart, Seascape architecture uses an open, industry-standard storage server that can scale up exponentially in both power and performance. Since the storage servers can integrate snap-in building blocks and software upgrades, you can quickly deploy new or improved applications and rapidly accommodate new data and media types. In this way, Seascape storage servers become an essential element for storing, manipulating, and sharing data across the network.

1.1.2 Enterprise Storage System overview

The IBM Enterprise Storage System can be configured in a variety of ways to provide scalability in capacity and performance. It employs integrated data caching (both volatile and non-volatile), hardware-level RAID5 support, and redundant systems at all levels. It provides easily configurable shared or secure access to user-variable quantities of storage via SCSI, Fibre Channel, ESCON or FICON I/O interfaces.

A single ESS unit can house up to 11 TB of usable protected storage. Up to 32 GB of cache can be installed in 8 GB increments.

ESS RAID5 storage is configurable by the customer via a convenient Web interface. It can be subdivided into logical disks which are then automatically configured to emulate disk device types that are compatible with the attached host computer systems. Because RAID5 storage is always striped in Enterprise Storage Servers, the I/O benefits of multiple active spindles and parallelism are immediately available; and RAID5 means all data is protected against device failure.

Multiple data paths - both internally and externally - can be specified to each logical disk created on an Enterprise Storage System. This multiplicity both enhances availability and increases I/O bandwidth. It also means that multiple host systems can be attached to either the same (shared) device, or to nonshared devices.

The combination of redundancy and hot-swappable components in the ESS translates to continuous availability. Whether it be a line cord or a critical data-filled disk, component failure is automatically handled without interruption of service. Enterprise Storage Systems have phone-home and error notification capabilities built-in.

Additional Web-based performance monitoring software is available for Enterprise Storage Systems; and a variety of standard and custom reports can be specified and scheduled for automated data collection.

1.1.3 ESS Copy Services components

Copy Services is a separately sold feature of the Enterprise Storage Server. It brings powerful data copying and mirroring technologies to Open Systems environments previously available only for mainframe storage. ESS Copy Services has two components: FlashCopy, and Peer-to-Peer Remote Copy (PPRC). Both help to off load backup tasks from your host systems.

Peer-to-Peer Remote Copy (PPRC) and FlashCopy are typically used as data backup tools for creation of test data and for data migration. They can also be used in disaster recovery scenarios.

Peer-to-Peer Remote Copy is a synchronous protocol that allows real-time mirroring of data from one ESS to another. In a disaster recovery scenario, this secondary ESS could be located at another site several kilometers away. PPRC is application independent. Because the copying function occurs at the disk subsystem level, the application has no knowledge of its existence. No host system processor is involved.

Copy Services provides both a Command Line Interface (CLI) and a Web-based interface for setting up and managing its facilities. The CLI allows administrators to execute Java-based Copy Services commands from a command line. The Web-based interface, a part of ESS Specialist, allows storage administrators to manage Copy Services from a browser-equipped computer.

Copy Services is of great use to customers with large IT systems, big data volumes, and a requirement for around-the-clock IS availability.

Copy Services will provide the most benefit to the customer who:

- Needs to have disaster tolerant IT centers
- Is planning to migrate data between systems
- Is migrating workload often
- Has to backup large amounts of data
- Needs to reduce the time the server has to be taken off-line for backup
- Plans to test new applications
- Needs a copy of production data for data warehousing or data mining

Copy Services can be integrated with technologies, such as Tivoli Storage Manager (TSM), formerly ADSM, Logical Volume Manager (LVM) mirroring, or SAN Data Gateway mirroring to solve a wide variety of business issues. IBM, with its broad portfolio of products in this industry, has many experts available to discuss the right solution for your business and to help you design and implement a solution that will give you the maximum business benefit.

FlashCopy

Because enterprise storage servers are highly intelligent subsystems, they are capable of performing storage-related activities independently of host computer systems. FlashCopy provides a point in time (PIT) copy of your data. This is known as T(0) copy. With FlashCopy, you can schedule and execute near-instantaneous copies of your data entirely within the ESS itself. Not only does this free your host processors for other activities, it also eliminates host-to-disk I/O normally associated with mirroring or other types of backup. If you look closely, you may notice some very slight performance degradation at the level of the logical disk being copied, but the rest of the system remains unaffected.

Peer-to-Peer Remote Copy (PPRC)

PPRC is essentially a synchronous mirroring activity utilizing two or more enterprise storage servers. No disk write to the source ESS is complete until its peers have also acknowledged the write. However, since enterprise storage servers can acknowledge writes once the data has been written safely to cache and nonvolatile RAM, PPRC does not require waits on remote drives. This means that PPRC is not burdened by the synchronous write latencies found in other mirroring systems.

Although PPRC is used primarily in high-availability or disaster recovery scenarios (remote systems can be located several kilometers away), it can also be used effectively for backups. Once a PPRC relationship has been established, the slave (also called a “clone”) will always track the master ESS. However, this tracking mechanism can be suspended long enough for a backup to be made, then resumed. This is called a “split-mirror” backup.

1.2 Introduction to DB2 UDB

DB2 Universal Database Version 7 (DB2 UDB) is IBM's object-relational database for UNIX, Linux, OS/2, and Windows operating environments. IBM offers DB2 Universal Database packages that provide easy installation, integrated functionality, a rich bundle of development tools, full Web enablement, OLAP capabilities, and flexibility to scale and change platforms.

1.2.1 DB2 Universal Database packaging

DB2 UDB (V7.2) for UNIX, Windows, and OS/2 environments is available in the following package options:

- DB2 Universal Database Personal Developer's Edition (PDE) provides all the tools for one software developer to develop desktop business tools and applications for DB2 Universal Database Personal Edition.
- DB2 Universal Database Personal Edition (PE) provides a single-user object-relational database management system for your PC-based desktop that is ideal for mobile applications or the power-user.
- DB2 Universal Developer's Edition (UDE) provides all the tools required for one software developer to develop client/server applications to run on DB2 Universal Database on any supported platform. Database servers and gateways can be set up for development purposes only.
- DB2 Universal Database Workgroup Edition (WE) a multi-user object-relational database for applications and data shared in a workgroup or department setting on PC-based LANs. Ideal for small businesses or departments.
- DB2 Universal Database Enterprise Edition (EE) a multi-user object-relational database for complex configurations and large database needs for Intel to UNIX platforms and from

uniprocessors to the largest SMP's. Ideal for midsize to large businesses and departments particularly where Internet and/or enterprise connectivity is important.

- DB2 Universal Database Enterprise-Extended Edition (EEE) provides a high performance mechanism to support large databases and offer greater scalability in Massively Parallel Processors (MPP's) or clustered servers. Ideal for applications requiring parallel processing, particularly data warehousing and data mining

All versions of the DB2 UDB work with the ESS. This redbook focuses on the EE and EEE packages.

1.2.2 The universal database

DB2 UDB is truly the universal database supporting a wide variety of platforms and applications.

Universal access

DB2 UDB provides universal access to all forms of electronic data. This includes traditional relational data as well as structured and unstructured binary information, documents and text in many languages, graphics, images, multimedia (audio and video), information specific to operations, such as engineering drawings, maps, insurance claim forms, numerical control streams, or any type of electronic information.

Access to a wide variety of data sources can be accomplished with the use of DB2 UDB and its complimentary products: Relational Connect, DB2 Connect, Data Joiner, and Classic Connect. Sources that can be accessed include: DB2 UDB for OS/390, DB2 UDB for OS/400, IMS, Oracle, MS/SQL Server, Sybase, NCR Teradata and IBM Informix databases.

Universal application

DB2 UDB supports a wide variety of application types. It is configurable to perform well for both on-line transaction processing (OLTP) was well as decision support systems (DSS). It can also be used as the underlying database for on-line analytical processing (OLAP).

DB2 UDB is also accessible from and/or integrated into a wide variety of Application Development environments. In addition to being able to call SQL functions from within standard programming languages, such as C/C++, COBOL, and Visual Basic, DB2 UDB fully supports Java technology and is accessible from Java applets, servlets and Applications. It also participates in Microsoft’s OLE DB as both a provider and a consumer.

Universal extensibility

The object-relational capabilities of DB2 UDB provide for its universal Extensibility. Through user defined functions (UDFs) and user defined types (UDTs) the base data types can be extended to include data types specific to your business. The framework of UDFs and UDTs provide the basis for DB2 UDB’s extensions in support of image, audio, video, text search, XML among others.

Universal scalability

DB2 UDB scales from pervasive / handheld devices with the DB2 Everyplace Edition, all the way to massively parallel processing environments (MPPs) with DB2 UDB Enterprise - Extended Edition (EEE). The various editions of DB2 UDB outlined above will run on the Palmtop, Laptop, Distributed Servers, Central Servers as well as clustered servers.

The superior scalability of DB2 UDB is made possible through a combination of features that are built into the base product. These include intra-partition parallelism as well as inter-partition parallelism for queries. The Database engine also takes advantage of I/O parallelism whenever possible, and features have been built-in to DB2 to recognize the unique characteristics of Disk I/O subsystems that use RAID-5, such as IBM's Enterprise Storage Server.

Universal reliability

DB2 UDB runs reliably across multiple hardware and operating systems. Major reliability features include transactional integrity through database locking as well as built-in Backup and Recovery capabilities. The scheduling of backups can be automated and with V7.2 incremental backups as well as full on-line or off-line backups can be taken. Backups can also be tailored to a single Tablespace. Recovery of a DB2 database can be done to a specific point in time, and can be filtered to only restore those database objects which are required to get you up and running.

Universal management

The DB2 Control Center includes a common integrated tool set to manage local and remote databases across all software and hardware client platforms from a single terminal. Components of the Control Center include:

- The Command Center provides a GUI window for inputting database or operating system commands, while allowing for storage, retrieval and browsing of previous commands.
- The Script Center allows for the creation, modification and execution of database or operating system scripts.
- The Journal Facility for managing jobs, recovery, alerts and messages.
- The Visual Explain facility provides a graphical means to display optimization-associated cost information and visual drill down of a query access plan.
- The Event Analyzer is a flexible tool for summary and historical performance analysis.
- The Performance Monitor supports online monitoring of buffer pools, sorts, locks, I/O and CPU activity.
- Smart guides guide database administrators through tasks, such as backup/recovery, performance configuration and object definition.
- The Alert Center displays objects which are in an exception status.
- The Index creation wizard helps database administrators build the best possible indexes for a given query workload.

All of the information presented in the GUI control center can also be accessed via a command line interface.

1.2.3 DB2 UDB optimizer

The query optimizer is the key component in the performance of any Enterprise Database Server. IBM has made more than a 25-year investment in enhancements to DB2's cost-based, rule-driven optimizer with the goal of keeping it the best in the industry. A major component of this effort is the implementation of the Starburst extensible optimizer in DB2 UDB. This allows IBM to add new intelligence to the optimizer without having to modify the entire query-compilation process.

As DB2 UDB is enhanced with new features and functionality, the optimizer performance improves. An example is extending the optimizer to understand OLAP SQL extensions and multiple levels of parallel query processing. The latter is particularly important in clusters where each node is an SMP. IBM has also built into the optimizer knowledge about how to work with underlying disk subsystems, such as the arrays of disks inside the ESS.

The optimizer takes advantage of its knowledge of the characteristics of the hardware environment; CPU speed, I/O speed, network bandwidth (for federated queries), bufferpool allocations. In addition, it knows the characteristics of the data itself: size of the table, partitioning scheme, uniqueness, existence of indexes, existence of automatic summary tables, when choosing an optimal execution strategy for a given SQL statement. This ability becomes more important as the size of the database increases.

In addition, the optimizer can also perform query re-write automatically. This enables DB2 UDB to execute better performing queries, while keeping the results set the same. This capability is especially important for environments where the SQL is being generated by a tool—true for many decision support applications.

1.2.4 DB2 UDB utilities

DB2 UDB utilities are designed to support both very large databases and also small OLTP databases as well. They incorporate a highly scalable software architecture which allows them to exploit a wide variety of hardware platforms. This book does not attempt to give a complete list of the DB2 utilities, we merely outline here those utilities which have the most interaction with the ESS. For more information on DB2 UDB utilities, please refer to *DB2 UDB Administration Guide: Performance V7*, SC09-2945.

Backup and recovery utilities

DB2 UDB provides a granular and parallel backup and restore utility. Some of the options available to backup include:

- Online or offline.
- Entire database, single tablespace or multiple tablespaces.
- For incremental backups, delta or cumulative backups can be taken.
- For partitioned databases, all partitions or a subset of the data partitions can be chosen.

The degree of parallelism achieved during the backup and restore process is determined by the number of backup devices available. This parallel capability results in a linear reduction in backup time.

For more information about the use of these utilities in an Enterprise Database Server, please refer to Chapter 7, “Database backup and recovery” on page 99.

DB2 UDB interfaces with storage management products, such as the IBM Tivoli Storage manager (TSM) product, which manages backup jobs and log archives from multiple servers. The TSM interface keeps track of backups and their associated logs. These products (integrated through the use of DB2 user exits) allow for the automated archival and retrieval of backups and log files to and from off-line storage.

Data movement utilities

There are some specialized utilities designed to help you move data into and within the database.

Load utility

DB2 UDB provides the ability to load large volumes of data into the database. The load utility bypasses the transaction logging mechanism, thus decreasing the elapsed time required to load data. The Load Utility utilizes multiple CPU engines in an SMP environment. It can execute in parallel by tablespace.

The DB2 UDB Load Utility supports the creation of indexes, backups and catalog statistics gathering (RUNSTATS) during the load execution. It also supports the ability to load data from remote systems.

Autoloader

In a partitioned environment, the autoloader utility is used to perform the necessary steps to load data from single or multiple flat files into a partitioned table. It splits data according to partition keys, it pipes the data to the appropriate partition, and concurrently loads data into each partition of the table. The Autoloader supports SMP parallelism within a partition as well as cross-partitions.

Data redistribution

For the DB2 UDB EEE partitioned database, the redistribute utility is used to move data between partitions when a new logical data partition is added or removed. It can also be used to redistribute data if data distribution across the existing data partitions is not uniform.

Table reorganization

The reorganization utility (REORG) re-clusters the rows of a table on disk in order according the a specified index. Clustering of each individual table provides enhanced performance. REORG can employ I/O parallelism when data is spread across multiple containers, and also executes in parallel for partitioned databases.

Export

The Export Utility allows for the extraction of data from the database in preparation for the movement of data from one database system to another, or to move from relational format to other formats for use in other applications, such as spreadsheet programs, among others.

Import

The Import Utility inserts data from an input file into a table or view.

DB2MOVE

The db2move utility allows data to be moved from one DB2 UDB database to another. These databases can reside on different servers, and can even be on different platforms.

DB2LOOK

The db2look utility can be used to obtain a point-in-time view of the characteristics of an existing database. It can extract the statistics as well as the required DDL to re-create the database on another server.

1.3 Introduction to AIX

AIX is the IBM variant of UNIX, and is a major player in Open Systems operating systems. Although AIX will be Linux compatible beginning with version 5. AIX Version 4.3, offers many enhancements over previous releases. It offers application binary compatibility, with very few exceptions, across all AIX Version 4 releases and when combined with the new 64-bit

hardware, introduces 64-bit computing to the pSeries and RS/6000 platforms for the first time. Its reliability and scalability, combined with 32/64-bit functionality, make it an ideal choice for todays heavy-duty commercial applications. AIX has all the features that you would expect from an industrial strength UNIX operating system. A few of its features are described below.

1.3.1 The Logical Volume Manager

The Logical Volume Manager (LVM) is often described as the heart of AIX. The Logical Volume Manager controls disk resources by mapping data between a simple and flexible logical view of storage space and the actual physical disks. The Logical Volume Manager does this by using a layer of device driver code that runs above the traditional physical device drivers. This logical view of the disk storage is provided to applications and is independent of the underlying physical disk structure.

Each individual disk drive is called a physical volume (PV) and has a name, usually **/dev/hdiskx** (where x is a unique integer on the system). Every physical volume in use belongs to a volume group (VG) unless it is being used as a raw storage device or a readily available spare (often called a 'Hot Spare'). Each physical volume consists of a number of disks (or platters) stacked one above the other. Each is divided into physical partitions (PPs) of a fixed size for that physical volume. For space allocation purposes, each physical volume is divided into five regions (outer_edge, outer_middle, center, inner_middle, and inner_edge), and these can be viewed as cylindrical segments cut perpendicularly through the disk platters. The number of physical partitions in each region varies depending on the total capacity of the disk drive.

The set of operating system commands, library subroutines, and other tools that allow the user to establish and control logical volume storage is called the Logical Volume Manager. The Logical Volume Manager controls disk resources by mapping data between a simple and flexible logical view of storage space and the actual physical disks. The Logical Volume Manager does this by using a layer of device driver code that runs above the traditional physical device drivers. This logical view of the disk storage is provided to applications and is independent of the underlying physical disk structure.

After installation, the system has one volume group (the root volume group called rootvg) consisting of a base set of logical volumes required to start the system plus any other logical volumes that were specified to the installation script. Any other physical volumes that are connected to the system can be added to this volume group (using the `extendvg` command) or used to create a new volume group (using the `mkvg` command). The following relationship exists between volume groups and physical volumes:

- On a single system, one to many physical volumes can make up a volume group.
- Physical volumes cannot be shared between volume groups.
- The entire physical volume becomes part of the volume group.
- The LVM is physical volume independent, thus, different types of physical volumes can make up a volume group within the limitation imposed by the partition size and partition limit.

Within each volume group, one or more logical volumes (LVs) are defined. Logical volumes are the way to group information located on one or more physical volumes. Logical volumes are an area of disk used to store data, which appears to be contiguous to the application, but can be non-contiguous on the actual physical volume. It is this definition of a logical volume that allows them to be extended, relocated, span multiple physical volumes, and have their contents replicated for greater flexibility and availability. Each logical volume consists of one or more logical partitions (LPs). Each logical partition corresponds to at least one physical

partition. If the logical volume is mirrored, then additional physical partitions are allocated to store the additional copies of each logical partition. These copies usually reside on different physical volumes (for availability) but, for performance reasons, may reside on the same physical volume.

1.3.2 Cache File System (CacheFS)

The Cache File System (CacheFS) is a general purpose file system caching mechanism available in AIX Version 4.3 that improves NFS server performance and scalability by reducing server and network load. Designed as a layered file system, CacheFS provides the ability to cache one file system on another. In an NFS environment, CacheFS can increase the client per server ratio, reduce server and network loads, and improve performance for clients on slow links, for example, PPP.

1.3.3 The Object Data Manager (ODM)

The Object Data Manager (ODM) manages the following system data:

- Device configuration data, both predefined (supported) and customized (installed)
- Software Vital Product data (SWVPD) (lists of products, history, inventory of files, and so on)
- System Resource Controller data (SRC)
- TCP/IP configuration data
- Error log and dump information
- Network installation manager (NIM) information
- SMIT menus and commands

Because the ODM is a database management system, its databases can be updated and queried easily and safely. Although the data is maintained in binary format to insure it is not carelessly corrupted, that information may be accessed by using the ODM command line interface or SMIT. This puts a wealth of otherwise dispersed information at the system administrator's fingertips.

Because the ODM maintains a status record of each item in its databases, software can be installed and applied for testing without committing it. This means that the applied software - including operating system updates - can be safely rolled back to the earlier version if problems should arise.

What's more, because the ODM maintains SMIT menus, the System Management Interface Tool can be updated or customized easily.

AIX has a plethora of commands not found in other versions of UNIX. Most of these commands are prefixed by the letters "`ls`." For the most part, these commands are simple command line abbreviations of predefined ODM database queries.

1.3.4 SMIT

The System Management Interface Tool, or SMIT, is a menu-driven interface to perform system administration tasks from within a consistent environment. SMIT is available in character, GUI (motif), and Web-based forms.

A truly wonderful feature of SMIT is that it does not perform any system management functions directly. It is a user interface that constructs high-level commands from the user's selections and then executes those commands on demand. These commands could be entered directly by the user to perform the same tasks.

This means that SMIT is a tremendous training, development, and empowerment tool. Because it dynamically constructs commands, the user can view these commands both before and after they are executed. Furthermore, SMIT automatically maintains a log and a script file of each user's activity. Not only is this a good record of system maintenance activity; it is also a very efficient means of preparing automated scripts. The user needs only to execute the function once from SMIT, then extract the command sequence from the smit.script file for inclusion in an automated shell script.

When using SMIT, the system configuration in ODM is maintained automatically. Most administrators never have a need to directly manipulate the ODM. SMIT and the high-level commands it issues ensure the ODM is correct.

2

Terminology and concepts

In this chapter we identify and define relevant ESS, AIX, and DB2 terminology. We discuss how a term from one system translates to the terminology of another; and we impart some clarity in situations when the same words are used with different meanings.

Upon occasion, we will mention how the topic is related to system performance, but on the whole we will defer a discussion of performance to a later chapter.

2.1 ESS concepts

The following describes some of the basic concepts and components of an ESS. Keep in mind that some of the terms may conflict with similar concepts within operating system or software layers. Even though we'll clarify things here, you'll find that some of the literature, especially earlier documents, will blur these definitions. Figure 2-1 shows some of the major components of an enterprise storage system.

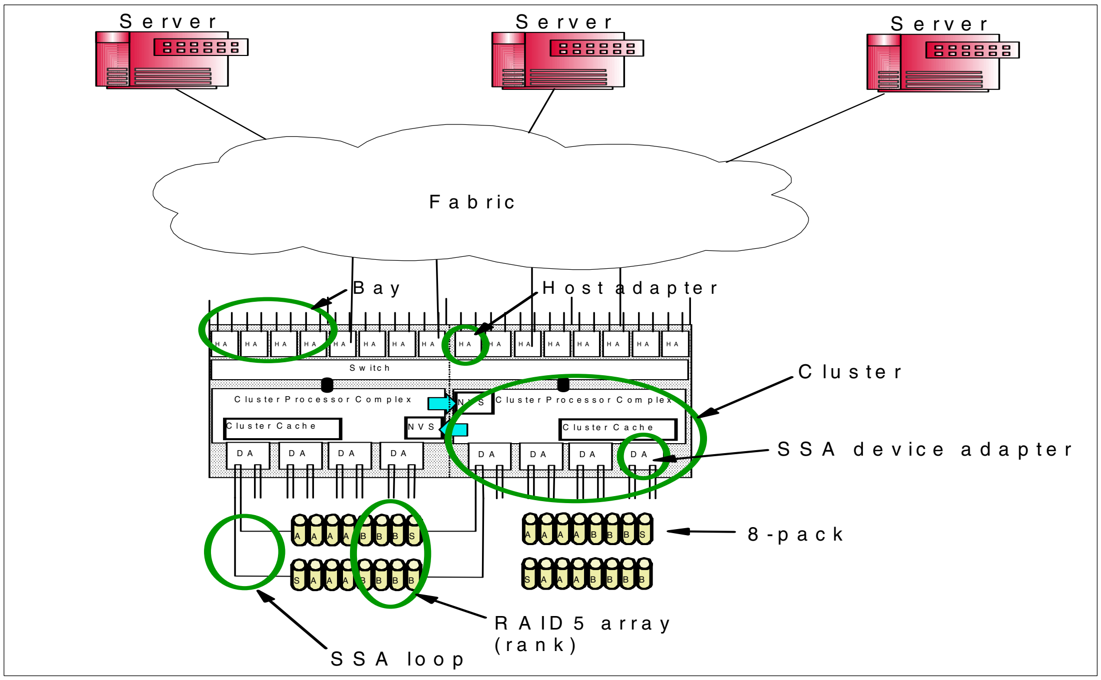


Figure 2-1 Enterprise Storage System components

2.1.1 Clusters

In the larger computer world, clusters are groupings of stand-alone systems (normally referred to as nodes), which have been networked together to provide high-availability or high performance configurations.

When talking about enterprise storage systems, a cluster is a collection of processors, cache, nonvolatile RAM (also known as nonvolatile storage, or NVS), and SSA disk adapters. Each ESS contains two such clusters. Each cluster normally controls one-half of the storage within the ESS, but can assume control of the other half if its partner should fail. Each cluster owns separate cache, disk adapters, and arrays, so under normal conditions does not affect performance of the other cluster.

These clusters run separate operating kernels, and communicate via an internal network connection. You can communicate with each cluster via the Web interface tool, ESS Specialist.

2.1.2 SSA disks, disk adapters, and RAID controllers

The ESS can contain up to 384 disks in a variety of capacities. Currently there are 9, 18 and 36 GB disk drives available. Although the same disk technology and capacities are available for SCSI, FC, FC-AL, ESCON and FICON attachment, ESS internally uses the Serial Storage Architecture (SSA) protocol for performing disk I/O operations. The SSA disk adapters (DA) provide almost all of the RAID functions within an ESS, such as parity calculations, disk rebuild and sparing, and so on. This provides performance benefits by off loading function from the central processors and cache.

The ESS hardware is configured in a *base frame* and *expansion rack*. The base frame of the ESS can hold up to 128 disk drives which are installed in up to sixteen 8-packs (see 2.1.4, “8-packs” on page 16). The base unit is divided into two cages, each cage holding up to eight 8-packs. The expansion frame can hold up to 32 8-packs, these are housed in 4 cages. Thus the expansion unit can hold up to a total of 256 disk drives. The layout of the disks drives can be seen in Figure 2-2.

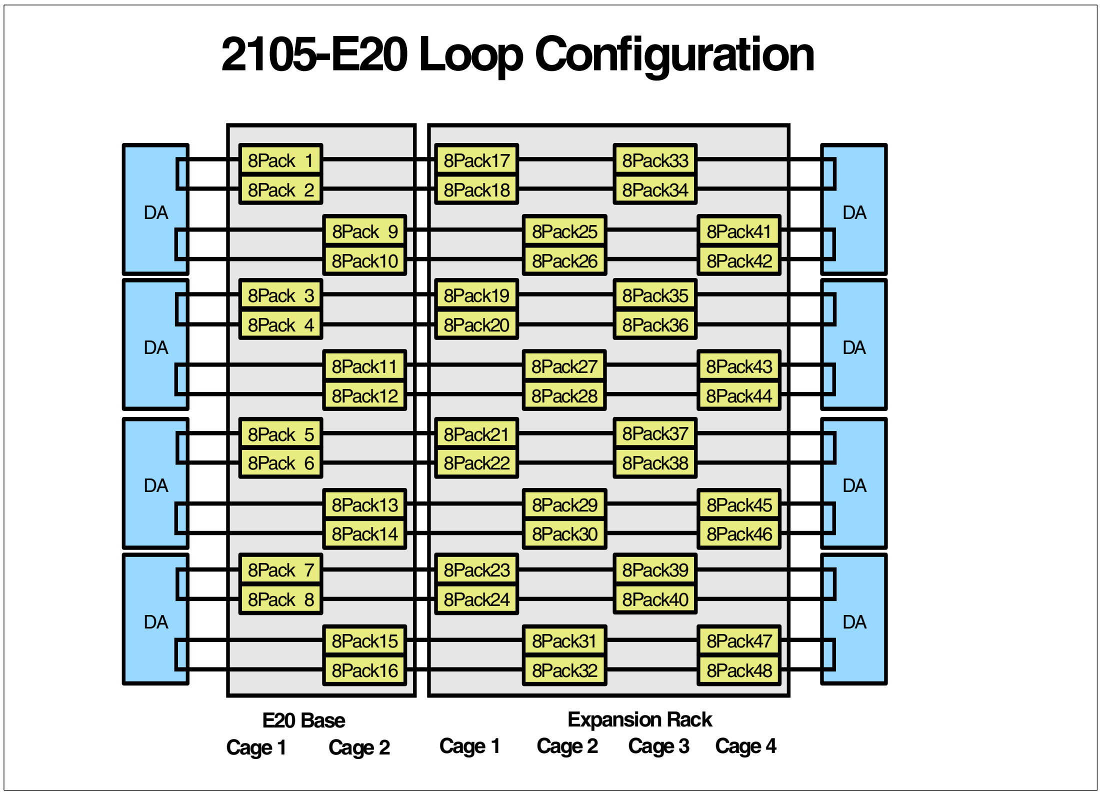


Figure 2-2 Schematic of layout of disk drives in ESS and expansion frame

2.1.3 SSA loops

SSA uses a non-arbitrated loop communication protocol, in which each disk adapter (DA) can access all devices on the loop. *Spatial reuse* allows data transfers from all of the devices to be multiplexed on the loop, allowing bandwidth to increase as disk adapters and disks are added to the loops. It also makes it possible to “twin-tail” devices, which means that two adapters can be connected to the same device.Enterprise storage systems use twin-tailing to provide high availability. If a cluster or device adapter should fail, the other cluster can access the storage from the twin adapter.

2.1.4 8-packs

An 8-pack is a physical collection of 8 disk drives, and disks are added to enterprise storage systems in multiples of 8-packs.

2.1.5 Arrays or ranks

An ESS array (also called a rank) is a collection of disks, with data striped across the disks for performance, and parity information stored to protect user data from disk failures. ESS configures its arrays using state-of-the-art RAID5 technology.

ESS provides RAID-5 arrays in sizes of either seven or eight disks. In addition, hot spare disks are reserved in case of disk failure. A RAID5 array is an array configured for high availability of data while minimizing disk usage. Data and parity information is striped across the disks in the array in such a way that if one disk were to fail completely, the data on that disk could be reconstructed on the spare disk using the data and parity information from the other disks of the array. This is commonly called a n-plus-parity configuration, where n is either six or seven disks.To avoid “hot spots,” parity information is striped across all disks, not localized on a single dedicated disk.

Two spare disks are provided within each SSA loop. Single-frame ESS configurations (which have 128 or fewer disks) contain 6+p arrays, with the remaining disks used as spares. When the disk system capacity is increased, 7+p arrays are added.

There is some performance degradation during the period of time after a disk fails and before the data is reconstructed on the spare disk, which is highly workload dependent. Striping data across multiple arrays can help minimize this impact.

Note: Do not confuse 8-packs (the physical packaging) with arrays. Arrays typically span two or more 8-packs.

2.1.6 Logical disks, LUNs and volumes

The storage administrator uses the ESS Specialist software to allocate storage as ESS logical disks. These logical disks appear to the operating system as physical disks, but in fact are logical storage allocated across the multiple physical disks in an ESS RAID array and do not span ESS RAID arrays. Logical disks are always created by “striping” across the array disks. This means that a section of disk space, called a “strip”, is claimed first from one disk, then the next, then the next, in a round-robin fashion until a logical disk of the required size has been built. ESS strips for fixed block devices (non S/390) are 32 kilobytes in size, so when data is read from or written to a logical disk, each 32 kilobytes transferred travels to or from a different disk.

Important: ESS logical disks are presented to the host operating system as if they were physical disk drives, such as hdisks. If there are multiple paths they are seen as vpaths. Thus, for example, an ESS logical disk appears as an AIX physical disk or volume. The terms used for the grouping of the ESS disk resources are very similar to terms used at the operating system level. When discussing resources at this level, it is important to keep in mind who’s point of view you are referencing.

Logical disk, LUNs and Volumes are all terms used to describe the resources outlined in Figure 2-3. This figure illustrates the construction of a logical disk by striping across an ESS RAID-5 array.

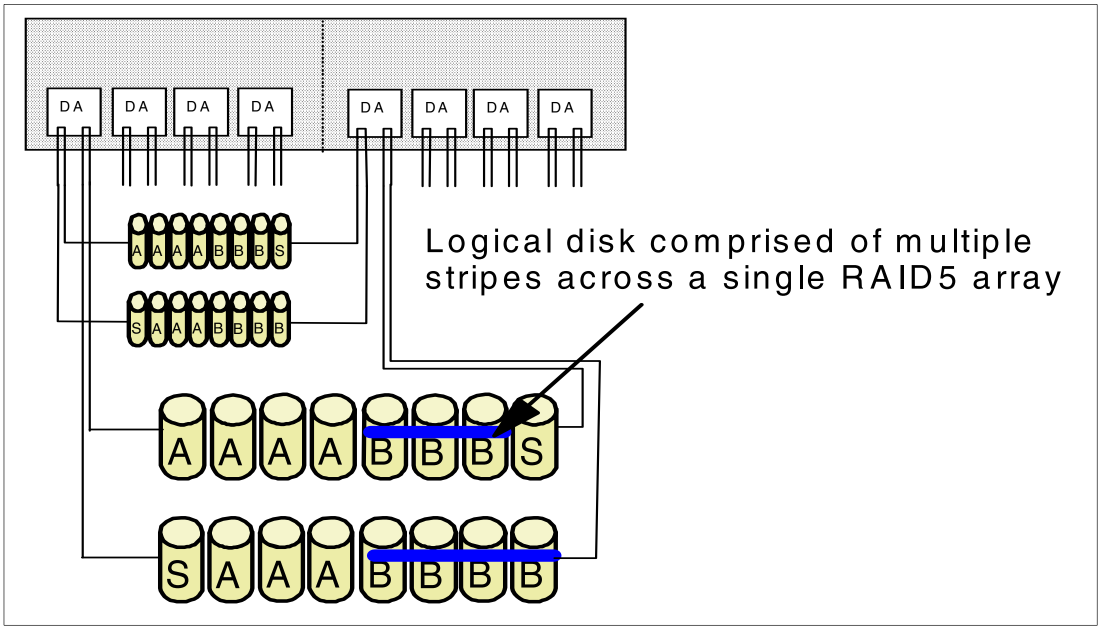


Figure 2-3 A logical disk within a RAID5 array

Logical disks are used to improve both manageability and performance. It is easier to manage a logical disk dedicated to a single function than to manage several smaller pieces of disks. Logical disks can also provide improved performance. especially in a striped RAID configuration. ESS supports many concurrent host accesses to a single logical disk, using all of the disks in an array simultaneously. Fibre channel and SCSI protocol supports this through command-tag-queuing, and OS/390 supports this through Parallel Access Volumes. It should be noted that the ESS does not provide any record or file locking if multiple hosts access the same logical disk. This is normally controlled by the application which will require a distributed lock manager or similar mechanism

Terminology considerations. ESS logical disks are sometimes called volumes (which can sometimes cause confusion with equivalent software terms, such as AIX or Veritas logical volumes). They are also sometimes erroneously called LUNs. LUN (logical unit number) was borrowed from SCSI terminology because an ESS logical disk will have a SCSI LUN address when Fibre channel/SCSI technology is used to connect the ESS to its host computer. This is the correct usage of the term LUN.

Note: Throughout this redbook, we will always use the term “logical disk” to identify the logical disk created as a series of stripes across a RAID5 array and presented to the host operating system as if it were a physical device.

2.1.7 Host adapters

Host adapters are used to connect the ESS data paths to host computers. They may be SCSI, UltraSCSI, Fibre Channel, ESCON or FICON adapters. Each has its own maximum bandwidth, which should be considered when calculating throughput to your logical disks. For open systems Fibre Channel gives the best performance.

The host adapters reside in *service bays,* four adapters per bay. Servicing an adapter requires swapping out an entire bay. Therefore, high-availability multipathing configurations should span at least two bays. For optimal performance, if there are two host adapters, bays 1 and 3, or bays 2 and 4 should be used.

2.1.8 Host attachment, LUN-masking, and multipathing

After a storage administrator creates logical disks, they can then use the ESS Specialist to specify which attached hosts and paths can access the logical disk though a technique called LUN-masking. For SCSI and Fibre-attached hosts, logical disks can be assigned as accessible through specific ESS host-adapter ports. For Fibre-Channel attached hosts, the administrator can also authorize host attachment according to the world-wide-name in the adapter in the host.

The storage administrator can make a logical disk available on more than one path, which might be connected to a single, or to multiple host systems. If the logical disk is available on more than one path in a single host system, multipathing software is used to control usage of those paths. The IBM Subsystem Device Driver (SDD) is one software component used to manage these multiple paths. SDD balances I/O traffic across those paths and diverts traffic to good paths in case of a path failure. Note that the ESS does not provide file or record locking. If multiple hosts are to access the same data it is the responsibility of the accessing application to provide locking.

In the case of AIX, each path to a logical disk appears to the operating system and system administrator as an *hdisk*. SDD recognizes that these separate hdisks represent a single logical disk, and constructs a logical entity (called a *vpath*) to represent that logical disk to the operating system. The AIX administrator then refers to the vpath when constructing AIX physical volume groups.

When an ESS is attached through a Fibre Channel switch, if switch zoning is not used, a logical disk might appear as an hdisk for every possible path combination. In other words, if a host system is attached through four ports on a switch and connected to four ports on an ESS, there are 16 possible path combinations, and hence 16 hdisks comprising the vpath.

In general a system with a smaller number of hdisks is easier to manage than one with a large number. Storage administrators may want to consider use of switch zoning, LUN-masking, and selection of larger logical disks to minimize the number of hdisks visible to each host system

2.1.9 Cache management

Each cluster manages cache storage (called cluster cache) and non-volatile storage (NVS.) The total cache ranges in size from 8GB to 32GB. Whenever an application reads from or writes to an ESS, a copy of the data is stored in the cluster cache. Subsequent read operations can then be satisfied from the cluster cache until the data “ages” from the cache. In addition, modified data is later destaged from the cluster cache to the RAID arrays.

Cache management strategies — reads

ESS is designed to manage its cache to fit the application’s data reference patterns:

- When an application references data very randomly, the ESS is likely to choose a record-caching strategy, meaning that only the requested physical record is brought into cache. In the case of DB2, this physical record usually corresponds to a DB2 page.
- In some cases, an application might frequently access a “nearby” physical record, in which case ESS adopts a “track-caching” strategy, meaning that ESS stages additional data into cache, corresponding to an ESS logical track. (An ESS logical track is the same as the physical strip size, roughly 32KB for SCSI and Fibre-Channel attached hosts.)
- If an application accesses data sequentially, as would likely be the case for database sequential scans, database physical backups, database load operations, and database

logging, then ESS recognizes this and adopts a sequential strategy. The sequential strategy incorporates the following actions:

- For sequential reads, ESS may prestage data into the ESS cache. This prestage operates asynchronously from host data requests, with the intent to have data available in the cache in time for the next request from the application. ESS usually prestages several RAID stripes, allowing multiple disks to work in parallel.
- ESS manages the cache storage to achieve a balanced efficiency between sequential and random access requests, preventing just a few sequential applications from dominating the entire cache storage. Sequential accesses tend to age more quickly from the ESS cache than random access requests.
- For sequential writes, ESS combines its cache and RAID management to make optimum use of the disk resources. It essentially converts RAID-5 sequential writes into RAID-3 operations. This is discussed more fully later.

Non-volatile storage

The NVS provides long-term protection for modified data in case of a cluster failure. The NVS for each cluster is physically maintained within the other cluster, so that the surviving cluster contains all of the cache-modified data in case of a cluster failure. When an application writes a record, ESS stores two copies of the record in cache (one in cluster cache and the other in NVS), and immediately indicates that the write is complete when both copies are stored safely. The record is doubly-protected at this point. Later, when the data is destaged from cache to disk (and therefore doubly-protected through RAID), the storage in NVS is available for other applications.

Managing sequential writes

Database logging can be a critical element for many workloads. The ESS design (hardware RAID, striping, and nonvolatile memory) work together for highly efficient logging operations. This also applies to any sequential writes, such as database loads. Here is how it all works.

- Data base logging usually consists of sequences of synchronous sequential writes. The patterns may not be purely sequential, meaning that they may periodically rewrite certain physical records, but they do have a generally sequential trend. All log writes will be of size 4K or multiples of 4K. The size of the actual writes depends on the work load and machine hardware. Log archiving functions (copying an active log to an archived space) also tend to consist of simple sequential read and write sequences.
- When the DBMS writes a log record ESS stores two copies of the log record in cache (one in cluster cache and the other in NVS), and immediately indicates that the write is complete when both copies are stored safely. The log-record is doubly-protected at this point.
- After several log records are written, ESS determines that this is a sequential write pattern. When data is moved asynchronously to disk, the cache manager sends one or more full-stripes of data to the SSA disk adapter.
- The ESS cache manager and disk adapter write data to each of the disks in parallel while at the same time internally calculating the parity and sending it to disk. This technique eliminates the traditional RAID-5 write penalty and mimics the operation of a RAID-3 writes.
- This technique can be more efficient than simple mirroring, RAID-1, or RAID 1+ 0. For any of those techniques, every megabyte of log data written requires 2 megabytes to be transferred to disk. For ESS, a megabyte of log data written requires approximately 1.16 (16% extra for parity in a 6+P array) megabytes to be transferred to disk. This means fewer disks, less disk or memory bus bandwidth consumed.

2.2 DB2 storage concepts

The database object which maps the physical storage to the database, is the Tablespace. Figure 2-4 illustrates how DB2 UDB is logically structured and how the tablespace maps to the underlying physical object(s).

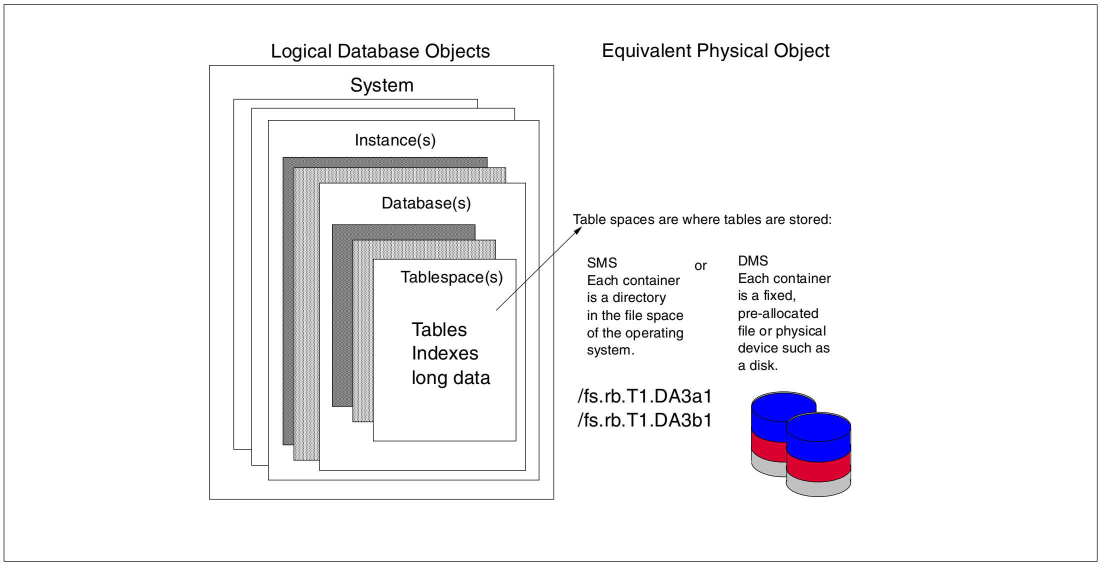


Figure 2-4 DB2 UDB logical structure

2.2.1 Instances

An instance is a logical database manager environment where you catalog databases and set configuration parameters. Another way of expressing the definition of a database manager instance is:

> A logical database manager environment similar to an image of the actual database manager environment. You can have several instances of the database manager product on the same database server. You can use these instances to separate the development environment from the production environment, tune the database manager to a particular environment, and protect sensitive information from a particular group of people.

For partitioned database systems, all data partitions will reside within a single instance.

2.2.2 Databases

A relational database presents data as a collection of database objects. The primary database object is the table (a defined number of columns and any number of rows). Each database includes a set of system catalog tables that describe the logical and physical structure of the data, configuration files containing the parameter values allocated for the database, and recovery log(s).

DB2 UDB allows multiple databases to be defined within a single database instance. Configuration parameters can also be set at the database level which allow the tuning of various characteristics, such as memory usage and logging.

2.2.3 Nodes and database partitions

A node number in DB2 UDB terminology is equivalent to a data partition. Databases which are comprised of multiple data partitions and which reside on an SMP system are also called multiple logical node (MLN) databases.

Nodes are identified by which physical system (CPU node) that they reside on as well as a unique node number. The node number, which can be from 0 to 999, uniquely defines a node. Node numbers must be in ascending sequence (gaps in the sequence are allowed).

The configuration information specific to the database as a whole is stored in the catalog node. The catalog node is the node from which you create the database.

Nodegroups

A nodegroup is a set of one or more database partitions. For non-partitioned implementations (all editions except for EEE), the nodegroup is always made up of a single partition.

Figure 2-5 shows how five nodes (data partitions) are divided into three different nodegroups. As you can see, one of the data partitions resides in multiple nodegroups.

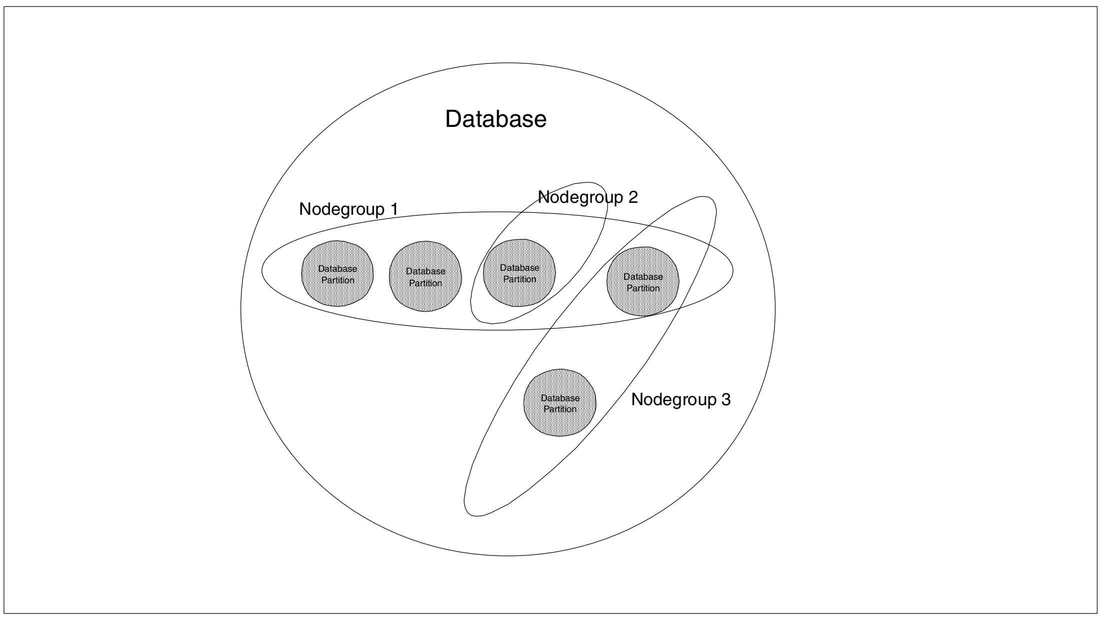


Figure 2-5 Nodegroups in a partitioned database

2.2.4 Partitioning map

When a nodegroup is created a partitioning map is associated with it. The partitioning map in conjunction with the partitioning key and hashing algorithm is used by the database manager to determine which database partition in the nodegroup will store a given row of data.

Partitioning maps do not apply to non-partitioned databases.

For more information on the partitioning capabilities of EEE, please refer to the Administration Guide, Chapter 4. Parallel Database Systems, or the DB2 UDB advanced certification guide for Clusters.

2.2.5 Containers

A container is a way of defining what location on the storage device will be made available for storing database objects. Containers may be assigned from file systems by specifying a directory. These are identified as PATH containers. Containers may also reference files which reside within a directory. These are identified as FILE containers and a specific size must be identified. Containers may also reference raw devices. These are identified as DEVICE containers, and the device must already exist on the system before the container can be used.

All containers must be unique across all databases; a container can belong to only one tablespace

2.2.6 Table spaces

A database is logically organized into parts called table spaces. A table space is a place to store tables. The table space is where the database is defined to use the disk storage subsystem. To spread a table space over one or more physical storage devices you simply specify multiple containers.

For partitioned databases, the table spaces reside in nodegroups. During the create tablespace command, the containers themselves are assigned to a specific node in the nodegroup, thus maintaining the 'shared nothing' character of DB2 UDB EEE.

Tablespaces can be either system managed space (SMS), or Data managed space (DMS). For an SMS table space, each container is a directory in the file system, and the operating system's file manager controls the storage space (LVM for AIX). For a DMS table space, each container is either a fixed-size pre-allocated file, or a physical device, such as a disk (or in the case of ESS, a vpath) and the database manager controls the storage space.

There are three main types of user table spaces: Regular (index and/or data), Temporary, and long. In addition to these user defined table spaces, DB2 requires a system table space, the catalog table space, to be defined. For partitioned database systems this catalog table space will reside on the catalog node.

2.2.7 Tables, indexes and LOBs

A table is a named data object consisting of a specific number of columns and some unordered rows. Tables are uniquely identified units of storage maintained within a DB2 tablespace. They consist of a series of logically linked blocks of storage that have been given the same name. They also have a unique structure for storing information that permits that information to be related to information in other tables

When creating a table you can choose to have certain objects, such as indexes and large object (LOB) data stored separately from the rest of the table data. In order to do this, the table must be defined to a DMS table space.

Indexes are defined for a specific table and assist in the efficient retrieval of data to satisfy queries. They can also be used to assist in the clustering of data.

Large objects (LOBs) can be stored in columns of the table. These objects, although logically referenced as part of the table, may be stored in their own table space when the base table is defined to a DMS table space. This allows for more efficient access of both the LOB data and the related table data.

2.2.8 Pages

Data is transferred to and from devices in discrete blocks that are buffered in memory. These discrete blocks are called *pages*, and the memory reserved to buffer a page transfer is called an I/O buffer. DB2 UDB supports various page sizes including 4k, 8k, 16K and 32k.

When an application accesses data randomly, the page size determines the amount of data transferred. In other words, it will correspond to the data transfer request size to ESS, which is sometimes referred to as the *physical record*.

Sequential read patterns can also influence the page size selected. Larger page sizes for workloads with sequential read patterns can enhance performance by reducing the number of I/Os.

2.2.9 Extents

An extent is unit at which space is allocated within a container of a table space for a single table space object. This allocation consists of multiple pages. The extent size (number of pages) for an object is set when the table space is created.

- An extent is a group of consecutive pages defined to the database.
- The data in the tables spaces is striped by extent across all the containers in the system.

2.2.10 Bufferpools

A bufferpool is main memory allocated in the host processor to cache table and index data pages as they are being read from disk, or being modified. The purpose of the bufferpool is to improve system performance. Data can be accessed much faster from memory than from disk; therefore the fewer times the database manager needs to read from or write to disk (I/O) the better the performance. Multiple buffer pools can be created.

2.2.11 DB2 prefetch (reads)

Prefetching is a technique for anticipating data needs and “reading ahead” from storage in large blocks. By transferring data in larger blocks, fewer system resources are expended and less total time is required.

Sequential prefetches read consecutive pages into the buffer pool before they are needed by DB2. List prefetches are more complex. In this case the DB2 optimizer optimizes the retrieval of randomly located data.

The amount of data being prefetched determines the amount of parallel I/O activity. Ordinarily the database administrator should define a prefetch value large enough to allow parallel use of all of the available containers, and therefore all of the ESS physical disks.

Consider the following example:

- A tablespace is defined with a page size of 16KB using raw DMS
- The tablespace is defined across four containers, and each container resides on a separate ESS logical disk, and each logical disk resides on a separate ESS RAID array.
- The extent size is defined as 16 pages (or 256KB)
- The prefetch value is specified as 64 pages (number of containers x extent size)
- A user issues a query that results in a tablespace scan, which then results in DB2 performing a prefetch operation.

The following would happen:

- DB2 would recognize that this prefetch request for 64 pages (a megabyte) evenly spans four containers, and would issue four parallel I/O requests, one against each of those containers. The request size to each container would be 16 pages, or 256KB.
- The AIX Logical Volume Manager would break the 256KB request to each AIX logical volume into smaller units (128KB is the largest), and pass them on to ESS as “back-to-back” requests against each ESS logical disk.
- As ESS receives a request for 128KB, if the data is not in cache, four arrays would operate in parallel to retrieve the data.
- After receiving several of these requests, ESS would recognize that these DB2 prefetch requests are arriving as sequential accesses, causing the ESS sequential prefetch to take effect. This will result in all of the disks in all four ESS arrays to operate concurrently, moving data into the ESS cache, all to feed the DB2 prefetch operations.

2.2.12 Page cleaners

Page cleaners are present to make room in the buffer pool before prefetchers read pages on disk storage and move them into the buffer pool. For example, if you have updated a large amount of data in a table, many data pages in the buffer pool may be updated but not written into disk storage (these pages are called dirty pages). Since prefetchers cannot place fetched data pages onto the dirty pages in the buffer pool, these dirty pages must be flushed to disk storage and become “clean” pages so that prefetchers can place fetched data pages from disk storage.

2.2.13 Logs

Changes to data pages in the buffer pool are logged. Agent processes updating a data record in the database update the associated page in the buffer pool and write a log record into a log buffer. The written log records in the log buffer will be flushed into the log files asynchronously by the logger. On UNIX, you can see a logger process (*db2loggr*) for each active database using the `ps` command.

Neither the updated data pages in the buffer pool nor the log records in the log buffer are written to disk immediately to optimize performance. They are written to disk by page cleaners and the logger respectively.

The logger and the buffer pool manager cooperate and ensure that the updated data page is not written to disk storage before its associated log record is written to the log. This behavior ensures that the database manager can obtain enough information from the log to recover and protect a database from being left in an inconsistent state when the database is crashed resulting from an event, such as a power failure.

2.2.14 Parallel operations

DB2 UDB extensively uses *parallelism* to optimize performance when accessing a database. DB2 supports several types of parallelism

- Query
- I/O
- Utility

Query parallelism

There are two dimensions of query parallelism: *inter-query parallelism* and *intra-query parallelism*. Inter-query parallelism refers to the ability of multiple applications to query a database at the same time. Each query executes independently of the others, but they are all executed at the same time. Intra-query parallelism refers to the simultaneous processing of parts of a single query, using either *intra-partition parallelism*, *inter-partition parallelism*, or both.

Intra-partition parallelism subdivides what is usually considered a single database operation, such as index creation, database loading, or SQL queries into multiple parts, many or all of which can be run in parallel within a single *database partition.*

Inter-partition parallelism subdivides what is usually considered a single database operation, such as index creation, database loading, or SQL queries into multiple parts, many or all of which can be run in parallel across multiple partitions of a partitioned database on one machine or on multiple machines. Inter-partition parallelism only applies to EEE.

I/O parallelism

When there are multiple *containers* for a tablespace, the database manager can exploit parallel I/O. Please reference **3.2.1, "Know where your data resides" on page 32** for the settings required when a container spans multiple physical disks. Parallel I/O refers to the process of writing to, or reading from, two or more I/O devices simultaneously; it can result in significant improvements in throughput.

DB2 implements a form of data striping by spreading the data in a tablespace across multiple *containers.* In storage terminology, the part of a stripe that exists on a single device is a "strip." The DB2 term for "strip" is *"extent."* If your tablespace has three containers, DB2 will write one extent to container 0, the next extent to container 1, the next extent to container 2, then back to container 0. The stripe width - a generic term not often used in DB2 literature - is equal to the number of containers, or three in this case.

Extent sizes are normally measured in numbers of DB2 pages.

For simpler disk systems, containers for a tablespace would ordinarily be placed on separate physical disks, allowing work to be spread across those disks, and allowing disks to operate in parallel. Since ESS logical disks are striped across an array, the database administrator can allocate DB2 containers on separate logical disks residing on separate ESS arrays. This will take advantage of parallelism both within DB2 and within ESS. For example, four DB2 containers residing on four ESS logical disks on four 7+P arrays would have data spread across 32 physical disks.

Utility parallelism

DB2 utilities can take advantage of intra-partition parallelism. They can also take advantage of inter-partition parallelism; where multiple database partitions exist, the utilities execute in each of the partitions in parallel.

The load utility can take advantage of intra-partition parallelism and I/O parallelism. Loading data is a CPU-intensive task. The load utility takes advantage of multiple processors for tasks, such as parsing and formatting data. It can also use parallel I/O servers to write the data to containers in parallel. Refer to the Data Movement Utilities Guide and Reference for information on how to enable parallelism for the load utility.

In a partitioned database environment, the Auto Loader utility takes advantage of intra-partition, inter-partition, and I/O parallelism by parallel invocations of the LOAD command at each database partition where the table resides. Refer to the Data Movement Utilities Guide and Reference for more information about the Auto Loader utility.

During index creation, the scanning and subsequent sorting of the data occurs in parallel. DB2 exploits both I/O parallelism and intra-partition parallelism when creating an index. This helps to speed up index creation when a CREATE INDEX statement is issued, during restart (if an index is marked invalid), and during the reorganization of data.

Backing up and restoring data are heavily I/O-bound tasks. DB2 exploits both I/O parallelism and intra-partition parallelism when performing backup and restore operations. Backup exploits I/O parallelism by reading from multiple table space containers in parallel, and asynchronously writing to multiple backup media in parallel. Refer to the BACKUP DATABASE command and the RESTORE DATABASE command in the *DB2 Command Reference for Common Server V2,* S20H-4645, for information on how to enable parallelism for these utilities.

2.3 AIX storage terminology

Figure 2-6 depicts the relationship between logical storage components in AIX

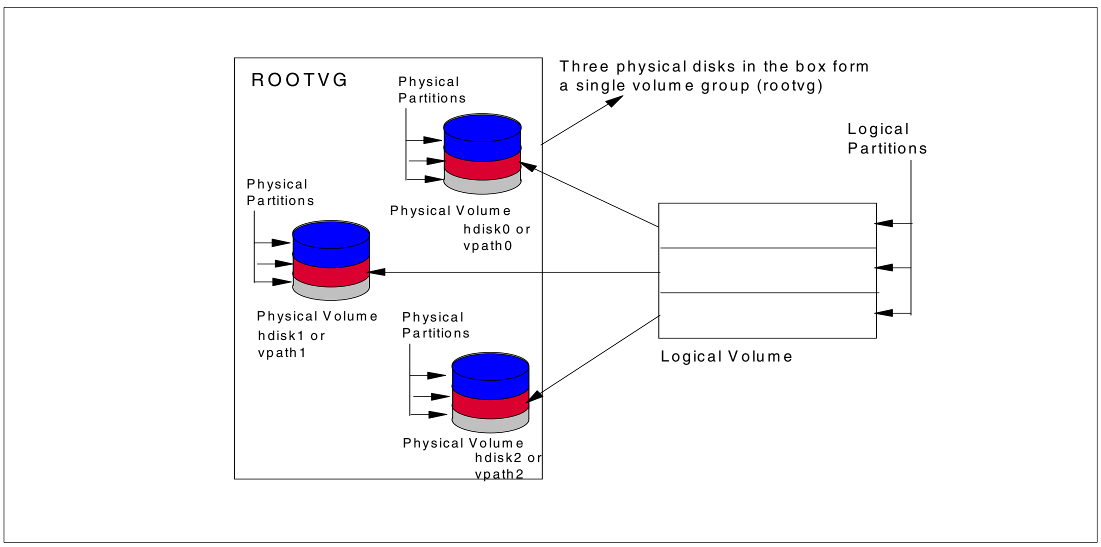


Figure 2-6 Relationship between logical storage components

2.3.1 Physical volume

To AIX, each individual fixed-disk drive is called a *physical volume* (PV) and has a name. Prior to the use of multi-pathing software, these physical volumes corresponded to specific hdisks. If virtual pathing is enabled, each hdisk represents one path to an ESS logical disk, and so the physical volume is represented by the vpath name.

Keep in mind that AIX physical volumes identifying ESS storage are not physical devices. They represent logical devices which are inherently striped across multiple disk drives.

2.3.2 Volume groups, logical volumes, and partitions

The AIX logical volume manager allows you to identify a collection of physical volumes collectively as a *volume group* (VG). Each physical volume will belong to a single volume group.

Volume groups can simplify storage management tasks, as they can be performed at the volume group level. For instance, volume groups can be exported from one system and imported into another; they can be backed up; and they can be removed altogether.

As a reminder, when using ESS as the underlying storage, it is possible to have more than one volume group spread across the same set of disk drives.

All of the physical volumes in a volume group are divided into *physical partitions (PPs)* of the same size.

Within each volume group, one or more *logical volumes* (LVs) are defined. Logical volumes are groups of information located on physical volumes. (One or more specific physical volumes can be specified when creating an LV). Data on logical volumes appear as contiguous to the user, but can be non-contiguous on the physical volume(s).

While physical volumes and volume groups are not normally addressed directly by users and applications to access data. Logical volumes provide the mechanism to make disk space available for use. Each logical volume consists of one or more *logical partitions* (LPs). Each logical partition corresponds to at least one physical partition. For more specific allocation polices see 3.3.1, “Logical volume mapping and striping” on page 38.

2.3.3 Journaled file systems

AIX’s native file system type is called the journaled file systems (JFS). Each journaled file system resides on a separate logical volume. The operating system mounts journaled file systems during initialization.

Filesystems are hierarchical structures built on logical volumes for storing data. They can also be connected together to form larger hierarchies by using the **mount** command. Filesystems use pointers for keeping track of individual files; and they have logs or journals. Both help reduce the amount of time required to access and modify files.

Filesystems are of fixed size; and they cannot share free space with other filesystems. Entire filesystems can be backed up or moved by the operating system.

Multiple journaled file systems use a common log, called a JFS log, configured to be 4 MB in size. For example, after initial installation, all file systems within the root volume group use logical volume hd8 as a common JFS log. The default logical volume partition size is 4 MB (for volume groups containing disks <= 4 GB), and the default log size is one partition, therefore, the root volume group normally contains a 4 MB JFS log.

Logical blocks

JFS divides the logical volume into a number of fixed size units called logical blocks. The first logical block (logical block 0) is reserved for a bootstrap program. The first and thirty-first are reserved for the superblock which contains information, such as the overall size of the file system in 512 byte blocks, the file system name, and so on.

Allocation groups

The rest of the logical blocks are divided into a number of allocation groups. An allocation group consists of data blocks and inodes to reference those data blocks when they are allocated to directories or files.

AIX files

AIX files are uniquely identified units of storage maintained within a file system. They consist of a series of logically linked blocks of storage that have been given the same name.

inode

The inode number is a unique number associated with each filename. This number is used to look up an entry in the inode table which gives information on the type, size, and location of the file and the userid of the owner of the file. When the file is opened by an application the inode is placed in a locked state and other applications cannot access the file until the file is closed by the first application releasing the lock. For in depth detail, please see the following Web site:

http://9.101.224.11/educ/kern_adv/adv_o.fm.html

3

Configuration for performance and manageability

This chapter discusses configuration issues for DB2 databases on IBM Enterprise Storage Server (ESS). The information presented here should take inexperienced architects 80% of the way towards an optimally configured system with the first 20% of their effort. It also identifies variations you may want to explore to further customize your database layout and tuning.

With the information presented in this chapter, you can bring a system up quickly with confidence that the configuration you have is good. With proper use of the tools at your disposal, you will be able to adapt that configuration to fit your needs even more closely.

3.1 Introduction

Enterprise storage systems are designed to make the job of managing storage easier, and at the same time meet the customers requirements for performance, availability, and cost. Determining the best configuration often requires balancing these sometimes-conflicting requirements.

Some configuration sizing decisions are made when a system is purchased. In the case of ESS, the most important of these decisions include:

- The total number of ESS disk systems used to support the application.
- The size of cache
- The amount of storage capacity
- The number of disks
- The number and types of host attachments (SCSI or Fibre Channel)

These decisions are discussed more fully in Chapter 4, "Sizing guidelines" on page 47**.** Once those configuration sizing decisions have been made, the customer is faced with a number of configuration options for DB2, AIX, and ESS. This chapter will guide you through some of those options and decisions.

3.1.1 Workload characteristics

In order to understand the performance issues associated with a particular database, it is helpful to have an understanding of the different database profiles and their unique workload characteristics.

While the characteristics of Online Transaction Processing (OLTP) and Decision Support Systems (DSS) in terms of disk usage can vary widely many of the recommendations in this book are valid for both workload types. When we feel that tuning or configuration parameters should vary due to workload type we will specifically state the differences. Otherwise it can be assumed that our advice applies to both.

Online Transaction Processing (OLTP)

OLTP databases are among the most mission-critical and widely deployed of any of the database types. Literally, millions of transactions encompassing billions of dollars are processed on OLTP systems around the world on a daily basis. The primary defining characteristic of OLTP systems is that the transactions are processed in real-time or online and often require immediate response back to the user. Examples would be:

- A point of sale terminal in a retail setting
- An Automated Teller Machine (ATM) used for withdrawing funds from a bank
- A telephone sales order processing site looking up inventories and taking order details

From a workload perspective, OLTP databases typically:

- Process a large number of concurrent user sessions
- Process a large number of transactions using simple SQL statements
- Process a single database row at a time
- Are expected to complete transactions in seconds, not minutes or hours

OLTP systems process the day-to-day operational data of a business and, therefore, have strict user response and availability requirements. They also have very high throughput requirements and are characterized by large amounts of database inserts and updates. They typically serve hundreds, if not thousands, of concurrent users, which can severely impact system performance.

Decision Support Systems (DSS)

DSS differ from the typical transaction-oriented systems in that they most often consist of data extracted from multiple source systems for the purpose of supporting end-user:

- Data analysis applications using pre-defined queries
- Application generated queries
- Ad-hoc user queries
- Reporting requirements

DSS systems typically deal with substantially larger volumes of data than OLTP systems due to their role in supplying users with large amounts of historical data. Whereas 100 gigabytes would be considered large for an OLTP system, a large DSS system would most likely be 1 terabyte or more. The increased storage requirements of DSS systems can also be attributed to the fact that they often contain multiple, aggregated views of the same data.

While OLTP queries tend to be centered around one specific business function, DSS queries are often substantially more complex. The need to process large amounts data results in many CPU intensive database sort and join operations. The complexity and variability of these types of queries must be given special consideration when designing a DSS system for performance.

3.2 Configuration considerations for ESS running with DB2

The following general principles will help you make the best use of an ESS disk system with DB2.

- Know where your data resides. Understand how DB2 containers map to ESS logical disks, and how those logical disks map to RAID arrays see Chapter 5, “Mapping ESS resources to AIX and DB2 UDB” on page 53. Spread DB2 data across as many RAID arrays as possible.
- Balance workload across ESS resources. Establish a storage allocation policy that allows balanced workload activity across RAID arrays. You can take advantage of the inherent balanced activity and parallelism within DB2, spreading the work for DB2 partitions and containers across those arrays.
- Use the inherent striping of DB2, placing containers for a tablespace on separate ESS logical disks which reside on separate ESS RAID arrays. This will eliminate the need for using underlying operating system or logical volume manager striping.
- Select an ESS logical disk size that allows for granularity and growth without proliferating the number of logical disks. Approximately eight logical disks per ESS RAID array works well for most environments.
- Use ESS multipathing along with DB2 striping to ensure balanced use of Fibre Channel or SCSI paths.

3.2.1 Know where your data resides

If you want optimal performance from ESS, don't treat it totally like a "black box." Establish a storage allocation policy that uses as many RAID arrays as possible. Understand how DB2 tables map to underlying logical disks, and how the logical disks map to RAID arrays. One way to simplify this process is to maintain a modest number of ESS logical disks. This is discussed in "Criteria for selecting ESS logical disk sizes" on page 35.

3.2.2 Balance workload across ESS resources

Use the inherent parallelism and balanced approach of DB2 to help balance I/O workload across the resources of ESS. This applies to both OLTP and DSS workload types. If you do that, and have planned sufficient resource, then many of the other decisions become secondary.

The following general principles can be used to your advantage:

- DB2 query parallelism allows workload to be balanced across CPU's and, if EEE is installed, across data partitions.
- DB2 I/O parallelism allows workload to be balanced across containers.

As a result, you can balance activity across ESS resources by applying the following rules:

1. Span ESS cabinets
2. Span clusters within a cabinet
3. Span disk adapters
4. Engage as many arrays as possible

Figure 3-1 illustrates this technique for a single table space consisting of eight containers.

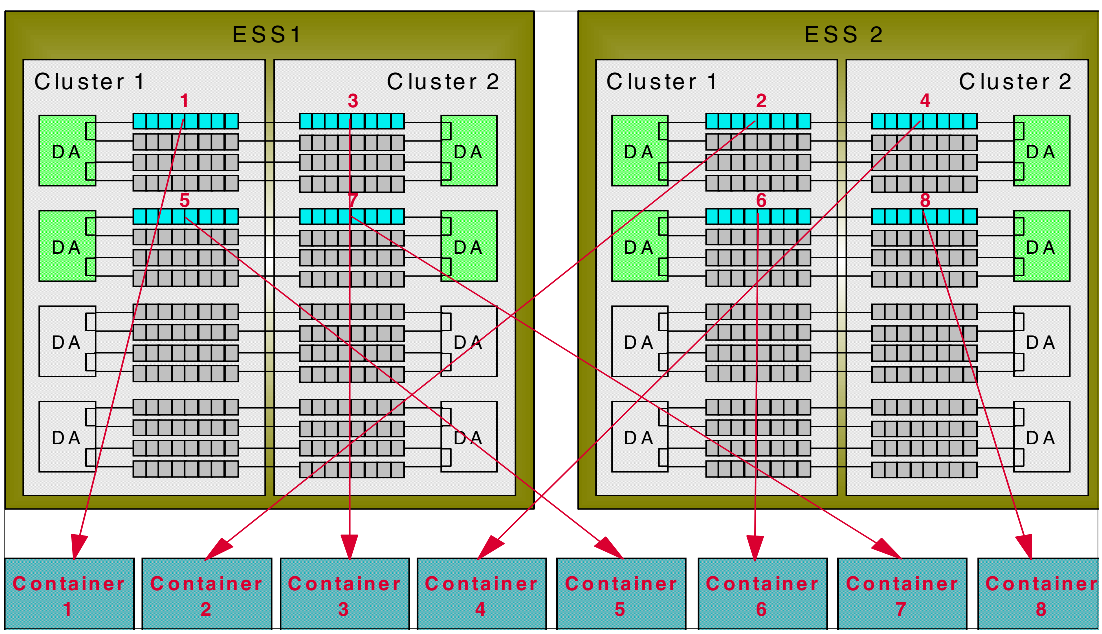


Figure 3-1 Allocating DB2 containers using a "spread your data" approach

In addition, the testing described in **Appendix A** substantiates the following points:

- You may intermix data, indexes, and tempspaces on RAID arrays. Your I/O activity will be more evenly spread and avoid the skew which you would see if the components were

isolated. This has the added advantage of making available more disk arms to each of these objects.

- For EEE systems, establish a policy that allows partitions and containers within partitions to be spread evenly across ESS resources. You can choose either a horizontal mapping, in which every partition has containers on every available ESS array, or a vertical mapping, in which DB2 partitions are isolated to specific arrays, with containers spread evenly across those arrays.
- For EEE systems, selection of horizontal or vertical mapping approaches may be influenced as you consider the number of DB2 partitions, the number of arrays, and how future growth will influence those factors. The vertical mapping approach works well as long as the number of ESS RAID arrays is an even multiple of the number partitions. Otherwise, the horizontal approach is probably best.

  Please see Appendix A, “Data placement tests” on page 107 for a full description of horizontal and vertical mapping.

Use DB2 to stripe across containers

Look again at Figure 3-1. In this case, we are striping across arrays, across disk adapters, across clusters, and across enterprise storage system boxes. This can all be done using the striping capabilities of DB2’s container and ‘shared nothing’ concept. This eliminates the need to employ AIX logical volume striping.

3.2.3 Selecting DB2 logical sizes

The three settings in a DB2 system that primarily effect the movement of data to and from the disk subsystem work together. These include page size, extent size, and prefetch size.

Page size

Page sizes are defined for each tablespace. There are four supported page sizes: 4K, 8K, 16K, and 32K. Some factors that affect the choice of page size include:

- The maximum number of records per page is 255. To avoid wasting space on a page, do not make page size greater than 255 times the row size plus the page overhead.
- The maximum size of a table space is proportional to the page size of its table space. In SMS, the data and index objects of a table have limits as shown in Table 3-1. In DMS, these limits apply at the tablespace level.

Table 3-1 Page size relative to tablespace size

Page size	Maximum data / index object size
4KB	64GB
8KB	128GB
16KB	256GB
32KB	512GB

For up to date limits, please refer to Appendix A.U, “SQL Limits” in the latest *DB2 SQL Reference for Common Server V2,* S20H-4665.

Select a page size that can accommodate the total expected growth requirements of the objects in the tablespace.

For OLTP applications that perform random row read and write operations, a smaller page size is preferable, because it wastes less buffer pool space with unwanted rows. For DSS applications that access large numbers of consecutive rows at a time, a larger page size is better, because it reduces the number of I/O requests that are required to read a specific number of rows.

Tip: Experience indicates that page size can be dictated to some degree by the type of workload. For pure OLTP workloads a 4 KB page size is recommended. For a pure DSS workload a 32 KB page size is recommended. And for a mixture of OLTP and DSS workload characteristics we recommend either an 8 KB page size or a 16 KB page size.

Extent size

If you want to stripe across multiple arrays in your ESS, then assign a logical disk from each array to be used as a DB2 container. During writes, DB2 will write one extent to the first container, the next extent to the second container, and so on until all eight containers have been addressed before cycling back to the first container. DB2 stripes across containers at the tablespace level.

The following are some factors in selecting these values:

- Since ESS stripes at a fairly fine granularity (32KB), selecting multiples of 32KB for extent size makes sure that multiple ESS disks are used within an array when a DB2 prefetch occurs.
- Tests conducted under this project have shown that I/O performance is fairly insensitive to selection of extent sizes, mostly due to the fact that ESS employs sequential detection and prefetch. For example, even if you picked an extent size, such as 128KB, which is smaller than the full array width (it would involve accessing only four disks in the array), the ESS sequential prefetch would keep the other disks in the array busy.

Tip: A good starting point would be to set extentsize equal to 1 complete stripe of disks in the LUN (for example, for a LUN striped across a 6+P array, set it to 6*32K = 196K

Prefetch size

The tablespace prefetch size determines the degree to which separate containers can operate in parallel.

Although larger prefetch values might enhance throughput of individual queries, mixed applications would generally operate best with moderate-sized prefetch and extent parameters.You will want to engage as many arrays as possible in your prefetch, to maximize throughput.

It is worthwhile to note that prefetch size is tunable. By this we mean that prefetch size can be altered after the tablespace has been defined and data loaded. This is not true for Extent and Page size which are set at Table space creation time and can not be altered without re-defining the table space and re-loading the data.

Tip: The prefetch size should be set such that as many arrays as desired can be working on behalf of the prefetch request. In non-ESS storage, the general recommendation is to calculate prefetch size to be equal to a multiple of the extent size times the number of containers in your tablespace. For ESS you may work with a multiple of the extent size times the number of arrays underlying your tablespace.

3.2.4 Selecting ESS logical disk sizes

ESS gives you great flexibility when it comes to allocating disk space. This is particularly helpful when you need to attach multiple hosts or implement security. However, this flexibility can present a challenge as you plan for future requirements.

Criteria for selecting ESS logical disk sizes

ESS supports a high degree of parallelism and concurrency on a single logical disk. As a result, performance tests have indicated that a single logical disk consuming an entire array achieves the same performance as many smaller logical disks on that array. However, consider how logical disk size affects systems management. Smaller logical disks allows for more granularity when managing storage, although it increases the number of logical disks seen by the operating system. Select an ESS logical disk size that allows for granularity and growth without proliferating the number of logical disks. If you are using FlashCopy you need to allow capacity in each LSS for the target volumes.

You should also take into account your container size and how the containers will map to AIX logical volumes and ESS logical disks. In the simplest situation the container, AIX logical volume and ESS logical disk will be the same size.

> **Tip:** Try to strike a reasonable balance between flexibility and manageability for your needs. Our general recommendation is that you create no fewer than two logical disks in an array, and minimum logical disk size should be around 16G. Unless you have an extremely compelling reason, use a single logical disk size throughout the ESS. Eight logical disks per RAID array can be a reasonable balance.

Smaller logical disks have the following attributes:

- Advantages:
  - They allow an administrator to assign storage to different applications and hosts
  - They allow greater flexibility in performance reporting. For example, ESS Expert reports statistics for logical disks.
  - They allow greater flexibility in use of Copy Services.
- Disadvantages:
  - Small logical disk sizes can contribute to proliferation of logical disks, particularly in SAN environments and large configurations.

Larger logical disks have the following attributes:

- Advantages
  - Simplifies understanding of how data maps to arrays.
  - Reduces the number of disks that appear to the operating system and storage administrators
- Disadvantages
  - Less granular storage administration

Examples:

Assuming a 6+P array with 36 GB disk drives. Suppose you wanted to allocate disk space on your 16-array ESS as flexibly as possible. You could carve each of the 16 arrays up into 8G logical disks, resulting in 26 logical disks per array (with a little left over). This would yield a total of 16 * 26 = 416 logical disks. Then you could implement 8-way multipathing, which in turn would make 8 * 416 = 3328 hdisks visible to the operating system.

Not only would this create an administratively interesting situation, but at every reboot the operating system would query each of those 3328 disks. Reboots could take a long time.

Alternatively, you could have created just 16 large logical disks. With multipathing and attachment of eight Fibre Channel ports, you would have 8 * 16 = 128 hdisks visible to the operating system. Although this number is large, it is certainly more manageable; and reboots would be much faster. Having overcome that problem, you could then use the operating system logical volume manager to carve this space up into smaller pieces for use. If you want to use FlashCopy then the maximum size of a logical disk is half the array size as you can only FlashCopy within the same Logical Subsystem (LSS).

There are problems with this large logical disk approach as well, however. If the ESS is connected to multiple hosts or it is on a SAN, then disk allocation options are limited when you have so few logical disks. You would have to allocate entire arrays to a specific host; and if you wanted to add additional space, you would have to add it in array-size increments. Furthermore, you could not utilize the full bandwidth of the ESS by striping across all arrays if only some arrays were accessible by a host.

This problem is less severe if you know your needs well enough to say that your ESS will never be connected to more than one host. Nevertheless, in some versions of UNIX an hdisk can be assigned to only one logical volume group. This means that if you want an operating system volume group that spans all arrays of the ESS, you are limited to a single volume group for the entire ESS.

DB2 can use containers from multiple volume groups, so this is not technically a problem for DB2. However, even if your ESS is primarily used by DB2, there will be some areas of storage that won't be (for example, staging data, OS files, and so on) So, if you want the ability to do disk administration at the volume group level (exports, imports, backups, and so on) then you will not be very pleased with a volume group that is three to eleven terabytes in size.

Finally, if you intend to use ESS Copy Services for high-availability or backup, you will need at least two logical disks per ESS logical subsystem.

If you decide to create thirteen 16G logical disks per array, then you will wind up with an additional 2G logical disk in each array. This 2GB is still usable and can be used for other purposes.

3.2.5 Employ ESS multipathing

Multipathing is the use of hardware and supporting software to provide multiple avenues of access to your data from the host computer. When using ESS, this means you need to provide at least two Fibre or SCSI connections to each host computer from any component being multipathed. It also involves internal configuration of the ESS host adapters and volumes.

Proper multipathing requires installation of the IBM subsystem device driver (SDD) software on your host computer to help keep track of everything. Without that software, one logical disk on multiple paths will appear as multiple disks to the operating system.

There are several benefits from using multipathing: high availability, high bandwidth, and ease of performance management. A high availability implementation is one in which your application can still access storage through an alternate path if a component on one path should fail. Multipathing for bandwidth means that you are supplying enough paths such that their total bandwidth does not limit the data throughput other components of the system can sustain. Multipathing for ease of performance management means allowing multipathing software to balance workload across the paths.

Multipathing for high availability

Components for high availability are built into enterprise storage systems. If a RAID5 disk fails, the data on that disk is either moved to a hot spare or recreated using surviving data and parity information on the other RAID5 disks. If the primary disk adapter on the primary cluster fails, then the secondary cluster takes control of the RAID5 array. For that matter, if any component on the primary cluster should fail, the secondary cluster automatically assumes control.

Part of the pathing within the ESS is configurable using the ESS Specialist interface. If you want to isolate your data for security or other reasons, you can specify isolated paths between each logical disk and its host computers; and, it is possible to specify a single path to a single host adapter.

For high availability, however, each volume should be pathed to at least two host adapters. These adapters should be located in separate bays of the ESS as well. Remember, a host adapter is not hot-swappable, and the bay must be powered off to replace failed adapters.

Multipathing for bandwidth and performance management

When you select the number of host attachments from any host system to ESS, you must first consider the bandwidth requirements for that host. Please refer to Chapter 4, “Sizing guidelines” on page 47.

When making logical disks available to a host, you can decide whether you want every logical disk to be available to every path, or whether you want to partition a set of paths to a set of logical disks. For example, if you have eight Fibre Channel paths directly connected to an ESS with access to 32 logical disks on 16 RAID arrays among other options, you could:

1. Allow all of the logical disks to be accessed by all eight Fibre Channel paths.
2. Subdivide the 32 logical disks into two groups, and allow four Fibre Channel paths to access each group.
3. Subdivide the logical disks in four groups, using two Fibre Channel paths each.

The characteristics of option 1 (above) are:

- Advantages:
  - It is the simplest to administer.
  - It allows the multipathing software to automatically balance workload across all of the available paths. This is particularly important when use of the logical disks may be unbalanced, causing imbalance in the use of the paths.
- Disadvantages:
  - It increases the number of hdisks; there will be eight hdisks for every ESS logical disk.
  - It increases the number of paths required for SDD path management to manage, and therefore somewhat increases software path length and increases processor utilization.

With proper configuration of DB2, and balanced activity to logical disks, it makes sense to configure logical disks in sets with approximately two or four paths. This is a reasonable balance of performance, availability, and ease of systems management.

Tip: We recommend at a minimum two host adapter connections to each logical disk, to satisfy both high-availability and bandwidth requirements. For high volume applications with lots of sequential reads, four host adapter connections are recommended.

3.3 AIX logical volume definitions

In this section we discuss AIX logical volume definitions and how to utilize them.

3.3.1 Logical volume mapping and striping

In most cases you should use DB2 striping to obtain best performance. In that case, a DB2 container corresponds to a single AIX logical volume, physical volume, and ESS logical disk. The containers (and therefore ESS logical disks) would all reside on separate ESS RAID arrays.

There may be a few exceptions to this simple data layout:

- If your ESS logical disks on an array are smaller than the desired container, then you can use AIX LVM to aggregate multiple physical volumes (and therefore ESS logical disks) into a single LV/container. Planning ESS logical disk sizes to match container sizes would avoid this.
- In some cases, you may choose to not use DB2 striping, for example if you decide that a single file system is much easier to manage, and that performance is secondary. In this case, you may still want to use some form of AIX LV striping to achieve better performance. There are two forms of striping available:
  - “Maximum spread allocation policy”. In this case you would define multiple PVs residing on separate ESS RAID arrays, and request that partitions be spread across multiple PVs with a “maximum spread” allocation policy. In this case, the “strip size” is the partition size. This can be thought of as a “coarse granularity” striping, and is often referred to as physical partition (PP) striping.
  - AIX LV striping. AIX allows LVs to be striped across PVs at a finer granularity than physical partitions. In this case, a strip size of 128KB might be appropriate.

When creating logical volumes, you specify the number of logical partitions for the logical volume, (the size of your logical partitions was determined when the volume group was created).

The size of your logical volumes really depends on how the logical volume is going to be used. If being used for DB2 UDB log files, the size and number of active log files needs to be calculated and the logical volume would need to be large enough to hold all of these plus a buffer. (In a partitioned database environment (EEE), these calculations are done at the partition level). Similar calculations need to be done whether the logical volume will be used for data, index, or temporary storage.

3.3.2 Logical volume mirroring

AIX logical volumes can also be mirrored across several physical volumes. For the most part, this is redundant with the RAID functions of ESS. However, you may consider its use in special circumstances, such as facilitating movement of data to ESS when migrating from other storage devices.

3.4 Other data placement considerations

Other factors that can be taken into account include the following.

Size of RAID arrays

ESS arrays come in two sizes: 6+P and 7+P, depending on the number of disks in the system. The performance-attentive among you might have realized that a 7+p array should perform better than a 6+p array. For writes and random I/O, you may see up to 15% difference between the two. For sequential applications, the differences are minimal.

As a general rule, try to balance workload activity evenly across RAID arrays, regardless of the size. It is not worth the management effort to do otherwise.

Outer edge or middle

With simple non-RAID disks, there are two placement factors that affect performance.

- Data placed on the outer edge can be transferred at a higher rate, because the disk surface spins faster on the outer diameters.
- Seek times are minimized for data that resides in the middle of the referenced data on disk. (this avoids frequently seeking to extreme edges of the disk).

In the case of ESS, the effect of the outer edge increasing data rate is minimal, due to the fact that data is striped across multiple disks in the array. The aggregate bandwidth of those disks exceeds the bandwidth of an array for data anywhere on the disk surface.

As a result, frequently-accessed files (such as logs) are best placed in the middle of the allocated space on an array, and therefore optimizing disk seek times. This would mean placing the logs on logical disks allocated in the middle of the array.

3.5 Data caching strategies

There are two place where we can utilize caching; in the host system using main memory and in the ESS cache. The main memory cache is mainly used by DB2 UDB as the area where buffer pools reside, ESS cache is automatically used by the system and is controlled by destaging and prefetch algorithms which are self tuning dependent on workload.

A buffer pool is an area of storage in memory into which database pages are temporarily read and changed. The purpose of the buffer pool is to improve database system performance by buffering the data in memory. Here data can be accessed from memory rather than from disk, so the database manager needs to read or write less to the disk. Not all data in DB2 is buffered; long field and LOBs are only accessed through direct I/O and are never stored in the buffer pool.

One component of the database manager is called Buffer Pool Services (BPS). The BPS is responsible for reading data and index pages from disk into memory and writing pages from memory to disk. BPS will use the File Storage Manager or the Raw Storage Manager to get the pages depending on whether an SMS or DMS table space is used. When creating a buffer pool, by default, the page size is 4 KB, but you can choose to have the page size set at one of these values: 4 KB, 8KB, 16 KB, or 32 KB. If buffer pools are created using one page size, then only table spaces created using the identical page size can be associated with them. You cannot alter the page size of the buffer pool following its creation.

Each database has at least one buffer pool (**IBMDEFAULTBP**), which is created when the database is created), and you can have more also. All buffer pools reside in the Database Global Memory which is available to all applications using the database. All buffer pools are allocated when the first application connects to the database, or when the database is explicitly activated using the **ACTIVATE DATABASE** command. Use this **ACTIVATE DATABASE** command to keep buffer pool primed even if all the connections terminate. This will be very useful when connection load is highly dynamic (for example, Web servers).

As an application requests data out of the database, pages containing the data are transferred to one of the buffer pools from disk. The next time an application requests data, the buffer pool is checked first to see if the data is there in the memory area; if it is found, BPS does not need to read data from the disk. Avoiding data retrieval from disk storage results in faster performance. If the buffer pools are not large enough to keep the required data in memory, the BPS has to read data from disk. Pages are not written back to the disk until the page is changed, or one of the following occurs:

- All applications disconnect from the database.
- The database is explicitly deactivated.
- The database quiesces (that is, all connected applications have committed)
- Space is required for another incoming page
- A page cleaner is available (NUM_IOCLEANERS database configuration parameter is not zero) and is activated by the database manager.

For further details on buffer pools please refer to the redbook *DB2 UDB V7.1 Performance Tuning Guide*, SG24-6012.

3.6 DB2 UDB tablespaces

How you set up your table space can have a large impact on your database performance. In **3.2.2, "Balance workload across ESS resources" on page 32** we outlined how you should spread your tablespace across as many ESS resources as you can. Here we discuss the various types of Tablespaces that you have to choose from.

Considerations

DB2 administrators can choose between several types of storage management options for tablespaces:

- Database-managed storage (DMS) raw
- DMS file
- System-managed (SMS)

The decisions for selection of the tablespace format are generally not affected by ESS. However, you might want to pay a little more attention to container layout for SMS tablespaces, particularly if those tablespaces are not spread across multiple containers on separate arrays.

The general considerations are as follows:

- DMS raw tablespaces generally provide the best performance for several reasons. First, the layers of file management are removed from the I/O path. Secondly, data generally maps fairly simply to underlying storage. Data that appears to DB2 as sequential access also appears to ESS as sequential access, making best use of caching algorithms. However, DMS is not as flexible in managing storage.
- SMS tablespaces provide the most flexibility in management of space. However, there are several performance considerations when using SMS.
  - Because file managers dynamically allocate disk storage, storage can become fragmented, and therefore become more difficult to maintain sequential access on disk. Tablespace scans might not appear to ESS as sequential accesses. As a result, it may be more important to ensure large extent sizes for data layout when using SMS tablespaces.
  - In some cases, because of file system designs, it can be more difficult to allow a high level of concurrent access to a single SMS container. If an SMS tablespace uses a small number of containers (for example, =1), it can be difficult to maintain the level of concurrent access that can be supported by the underlying ESS array and the host operating system. As a result, if using SMS, be sure to allocate a sufficient number of containers on a single RAID array, particularly if you have few arrays and if database load performance is especially important. We recommend four to eight SMS containers for every ESS RAID array, depending on how many arrays will be used. (for example, if a single array is to be used, then we recommend eight containers, however if using more than three arrays then four containers per array may be sufficient.)

3.6.1 SMS tablespaces

SMS stands for system managed space. This means that DB2 does nothing to manage the space assigned to the tablespace. The operating system owns the directory, creates, grows, shrinks, and removes the files within it, and controls all of the I/O.

Most DB2 databases use SMS tablespaces for temporary storage. That way, the operating system allocates space only as it is needed; and at some point, the size of temporary storage reaches grows and shrinks. You get the most inexpensive solution in the safest way; and unused space in the filesystem remains available for other purposes.

On the negative side, when SMS tablespaces grow dynamically, they compete for system resources at a time when those resources are in high demand. Also, you can't add containers to an SMS database dynamically like you can DMS tablespaces. The only way to add containers is through a backup and redirected restore.

There is also a performance penalty to be paid for using SMS tablespaces. First, system I/O buffers are used in addition to the DB2 buffers already allocated. Not only is this an additional layer in the I/O path, but DB2 must compete dynamically with other applications for system I/O buffer resources.

Another performance penalty is paid by the system journaling of I/O activities and subsequent flushing of that journal to disk. Journaled file systems have come a long way in the past few years, and that penalty is not as severe as it used to be, but it's still present. More importantly, this journaling is more or less duplicated by DB2 logs.

All in all, the conventional wisdom is that you will pay approximately a 10% performance penalty for choosing SMS tablespaces. Many people are willing to pay that penalty for the convenience.

3.6.2 DMS raw tablespaces

DMS stands for database managed storage. This represents the other extreme. Although the operating system knows that the storage devices exist, that storage is not made available to the system. It is given over to DB2 “raw,” unsullied by any operating system intervention.

In tablespaces composed of raw containers, there is no file system overhead. DB2 creates, grows, shrinks, and deletes tables; allocating and freeing space within the container. Since container space is preallocated, DB2 does not have to compete for system resources as it manages this space. We’ve already mentioned that table objects, such as Indexes and LOBs can reside in separate DMS tablespaces, and that containers can be added to them or extended.

Furthermore, no system overhead is expended in preparing and launching an I/O in addition to that already done by DB2. The data remains grouped in extents; and it is buffered in memory reserved exclusively for DB2. The system I/O layer is bypassed.

DMS raw tablespaces are the best performing, especially for OLTP workloads and large DSS reads, such as applications which insert into append tables. If you can get past the initial discomfort of having a database that is not visible to the operating system except through DB2 tools, they really aren’t any harder to manage.

Here are the drawbacks: DB2 raw tablespaces are limited to the total size of the containers assigned to them; unused space in those containers cannot be shared.

3.6.3 DMS file tablespaces

DMS file tablespaces are a compromise between SMS and DMS raw tablespaces. In this case, filesystem files are created and then given to DB2 to manage. DB2 handles the files as if they were raw devices.

The drawbacks are that there is a little system overhead in maintaining the container files; the tablespaces are limited to the total size of the files assigned; and unused space cannot be shared. The benefits are improved performance over SMS tablespaces, and the ability to perform conventional filesystem backups.

3.6.4 Tablespace definition parameters

For SMS managed table spaces it is necessary to define all of the containers (directories) that you will be requiring when you create the table space. Containers can not be added or deleted after an SMS table space has been created. (Storage is altered using the operating system methods, for AIX using LVM).

For DMS table spaces, containers are also specified at table space creation time, however, containers can be altered (increased in size) and additional containers can be added to increase space using an ALTER TABLESPACE command.

The following figure, Figure 3-2 on page 43, outlines the configurable parameters set during the creation of the table space.

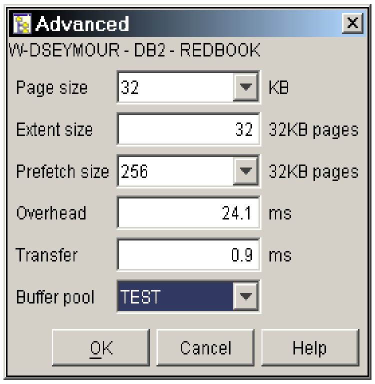


Figure 3-2 Table space configuration parameters

Page size, Extent size and Prefetch Size were discussed previously in "Selecting DB2 logical sizes" on page 33. Following are descriptions of the other required parameters.

Overhead and transfer rates

DB2 UDB defines two parameters related to I/O operational performance: *overhead* and *transfer rate*. These parameters are taken into account when making optimization decisions, and help determine the relative cost of random versus sequential accesses.

Overhead

OVERHEAD, which provides an estimate (in milliseconds) of the time required by the container before any data is read into memory. This overhead activity includes the container's I/O controller overhead as well as the disk latency time, which includes the disk seek time.

The default value for overhead is 24.1 milliseconds. This is the number we used during our testing.

Transfer rate

TRANSFERRATE, which provides an estimate (in milliseconds) of the time required to read one page of data into memory.

The following values (Table 3-2) are recommended for use with ESS.

Table 3-2 Suggested overhead and transfer rates

Overhead	**12.5**
Transfer rate (4KB)	**0.1**
Transfer rate (8KB)	**0.3**
Transfer rate (16KB)	**0.7**
Transfer rate (32KB)	**1.4**

3.7 DB2 and system configuration parameters

There are many parameters associated with the DB2 UDB relational database system. We will present here the ones that are mainly concerned with the disk I/O operations, highlighting those which are of particular interest to ESS.

DB2 UDB includes two type of configuration files: the database manager configuration file (set at the instance level), and the database configuration file for each database. In addition to these two configuration files, additional tuning capabilities are introduced as DB2 registry variables. These are entered using a db2set command.

It is worthwhile to note that DB2 UDB provides a Smart Guide for setting these configuration parameters. The Smart Guide is accessible from the Control Center, and will give you a good starting point from which to tune your database. It is recommended that every database you use start out with the recommendations from the smart guide rather than leaving the installation defaults in place.

When using DB2 on Enterprise Storage Systems, parallelism is a key factor. Pay particular attention to any mechanism for enhancing parallelism. In particular, be sure to use the following parameters or know why you aren't using them.

3.7.1 DB2 registry variables

The following registry variables are involved in the interaction of the ESS and DB2.

DB2_PARALLEL_IO

When reading data from, or writing data to table space containers, DB2 may use parallel I/O if the number of containers in the database is greater than 1. However, there are situations when it would be beneficial to have parallel I/O enabled for single container table spaces. For example, if the container is created on a RAID device that is composed of more than one physical disk, you may want to issue parallel read and write calls.

To force parallel I/O for a table space that has a single container, you can use the DB2_PARALLEL_IO registry variable. This variable can be set to "*" (asterisk), meaning every table space, or it can be set to a list of table space IDs separated by commas. For example:

```
db2set DB2_PARALLEL_IO=*        {turn parallel I/O on for all table spaces}
db2set DB2_PARALLEL_IO=1,2,4,8  {turn parallel I/O on for table spaces 1, 2, 4, and 8}
```

After setting the registry variable, DB2 must be stopped (db2stop), and then restarted (db2start), for the changes to take effect. The registry variable also affects tablespaces with more than one container defined. Without the registry variable set, the I/O parallelism is equal to the number of containers in the tablespace. With the registry variable set, the I/O parallelism is equal to the result of (prefetch size / extent size). One might want to do this if the individual containers in the tablespace are striped across multiple physical disks.

For example, a tablespace has two containers and the prefetch size is four times the extent size. If the registry variable is not set, a prefetch request for this tablespace will be broken into two requests (each request will be for two extents). Provided that the prefetchers are available to do work, two prefetchers can be working on these requests in parallel. In the case where the registry variable is set, a prefetch request for this tablespace will be broken into four requests (one extent per request) with a possibility of four prefetchers servicing the requests in parallel.

In this example, if each of the two containers had a single disk dedicated to it, setting the registry variable for this tablespace might result in contention on those disks since two prefetchers will be accessing each of the two disks at once. However, if each of the two containers was striped across multiple disks, setting the registry variable would potentially allow access to four different disks at once.

DB2_STRIPED_CONTAINERS

When creating a DMS table space container a one-page tag is stored at the beginning of the container. The remaining pages are available for storage by DB2 and are grouped into extent-size blocks of data.

When using RAID devices for table space containers, it is suggested that the table space be created with an extent size that is equal to, or a multiple of, the RAID stripe size. However, because of the one-page container tag, the extents will not line up with the RAID stripes, and it may be necessary during the I/O request to access more physical disks than would be optimal. This is particularly important when the RAID devices are not cached, and do not have special sequential detection and prefetch mechanisms (such as native SSA disks).

When DB2_STRIPED_CONTAINERS is set to ON, when the tablespace is created, the initial container tag will take up a full extent. This will avoid the problem described above.

> **Note:** We recommend that DB2_STRIPED_CONTAINERS is set to on. With ESS, sequential detection and prefetch may alleviate this problem.

3.7.2 Database manager configuration settings

Since the mapping of the database to the underlying storage device is done at the Table space level, and table space resides in a particular database. There are no database manager configuration settings which apply directly to storage.

3.7.3 Database configuration settings

The following database configuration settings are involved with the interaction of the ESS and DB2. This is not meant to be a comprehensive list of tunable database configuration parameters. For a complete listing of available parameters, please refer to the *DB2 Command Reference for Common Server V2,* S20H-4645.

CHNGPGS_THRESH — Changed pages threshold. Used to specify the level (percentage) of changed pages at which the asynchronous page cleaners will be started, if they are not currently active.

DFT_EXTENT_SZ — Default extent size of table spaces (in pages).

DFT_PREFETCH_SZ — Default prefetch size of table spaces (in pages).

LOGFILSIZ — Specifies the amount of disk storage, in pages, allocated to log files used for data recovery. This parameter defines the size of each primary and secondary log file.

NUM_IOCLEANERS — Specifies the number of asynchronous page cleaners for a database.

NUM_IOSERVERS — Specifies the number of I/O servers for a database. I/O servers are used on behalf of the database agents to perform prefetch I/O and asynchronous I/O by utilities, such as backup and restore.

SEQDETECT — Indicates whether sequential detection for a database is to be enabled or disabled.

3.7.4 Enable multi-page file allocation

SMS table spaces are expanded on demand. This expansion is done a single page at a time by default. However, in certain work loads (for example, when doing a bulk insert) you can increase performance by using the db2empfa tool to tell DB2 to expand the table space in groups of pages or extents. The db2empfa tool is located in the bin subdirectory of the sqllib directory. Running it causes the multipage_alloc database configuration parameter to be set to YES. For more information on this tool, refer to the *DB2 Command Reference for Common Server V2,* S20H-4645.

4

Sizing guidelines

This chapter is designed to help you understand the issues relating to making sure you have the right sized ESS and Enterprise Server for your Enterprise Database Server needs.

The following general topics are discussed:

- Understanding your requirements
  - Capacity and growth
  - Throughput and transaction rates
- How to size the disk and server
- Rules of thumb
- Example scenario

The objective of sizing a system should be a balanced system. A balanced system is a system that has a sensible set of CPU, memory, disks, and network connections. There should be no bottleneck which would make the other components ineffective.

We will be focusing on the ESS requirements in this chapter, including information about the sizing of the Enterprise Server, where appropriate.

4.1 Understanding your requirements

It is essential to understand the requirements of your system in order to obtain good performance. Determining the size of ESS that you will need is dependent not only on capacity, but on throughput and availability as well. Capacity should only be considered a starting point in determining how many arrays and ESS systems you will need.

4.1.1 Capacity and growth

How do you know what capacity is required? Capacity is generally a simple mathematical calculation. You start with information about the base data that will need to be stored for a particular application, and then add to that for indexes, temporary space, summaries and replicated tables, logging and data staging areas.

The simplest calculation involves knowing the length of the data and indexes per row of your table. A spreadsheet can be set up which simplifies this task.

Table Name	Row Size	# of Rows (M)	Total Raw data size (Gb)
Customer	900	20	18.0
Item	300	1	0.3
Sales	50	5000	250.0
...			
...			
...			
Total			268.3

Figure 4-1 Calculating raw data requirements

Once the Total raw data is known you can continue with educated sizing calculations to complete your capacity estimates, or you can apply some general rules of thumb.

Rules of thumb

Rules of thumb can be employed to move you from the base data (also referred to as raw data) to the entire amount of space required. In the past, this rule of thumb has been 3 to 1, for example, for every 1 GB of raw data, you would need 3 GB in storage. While this rule of thumb is useful, you may want a more precise method of calculating your capacity requirements. (Especially as the cost of storage may be the largest portion of your budget for this system).

Using Example 4-1, the rule of thumb would indicate a minimum of 805 GB of storage as a starting point.

Calculations

To calculate your capacity requirements more precisely, you will need to know more about your planned database, including how many indexes are defined for each table, the length of the indexes as well as similar information for summary tables. This information combined with the expected number of rows of the tables will give you a good start on calculating your capacity requirements.

The following example shows another spreadsheet calculation for base storage. The result in this case is slightly higher than what the Rules of Thumb method would indicate but should be more accurate.

Table Name	Row Size	# of Rows (M)	Total Raw data size (GB)	
Customer	900	20	18.0	
Item	300	1	0.3	
Sales	50	5000	250.0	
...				
Total			268.3	
Table Name	**Index Name**	**Row Size**	**# of Rows (M)**	**Total Table Size (GB)**
Customer	Name	40	20	0.8
Customer	zipcode	10	20	0.2
Sales	Sales_IX	15	5000	75.0
Sales	Sales_name_IX	15	5000	75.0
...				
Total				151.0
Summary Table Name	**Base table**	**Row Size**	**# of Rows (M)**	**Total Table Size (GB)**
Sales by product	Sales	50	1000	50.0
Sales by rep	Sales	50	100	5.0
...				
Total				55.0
Raw Data	**268.3**			
Indexes	**151.0**			
Summaries	**55.0**			
Workspace	**142.3**			
Overhead	**41.9**	**(row headers, disk formatting)**		
Disk overhead	**164.6**	**(25% for disk protection)**		
Total space	**823.2**			
Total space	**823.2**			
Growth factor	0.45			
New space req'd	**1193.6**			

Figure 4-2 Calculating base total storage requirements

Room for growth

While the rules of thumb and calculations will let you know how much ESS is required at a minimum to store the data you expect to have, we have not yet taken into account growth requirements.

Growth must take into account the following types of questions:

- How many new customers/records do we expect to have over the next year/month/quarter?
- How long do I expect this system to last before I will upgrade hardware?
- Are there going to be changes in my business that will require me to track additional data?

Most systems are expected to grow by a factor from 30 percent to 100 percent per year.

Once these questions are answered, then the base capacity you have calculated should be modified to reflect the growth expected over the time period you expect the system to last.

	Growth Rate	Capacity
Base capacity		**823.2**
Year 1	0.3	1070.1
Year 2	0.5	1605.1
Total		**1070.1**

Figure 4-3 Capacity plus growth calculation

Additional considerations for capacity

Other considerations for capacity include understanding whether or not you will be using some of the advanced capabilities of the ESS, such as Copy Services. In the calculation examples above, we included space for disk protection (the disk overhead calculation from Figure 4-2) but not for Copy Services.

4.1.2 Throughput or transaction rates

Simply knowing how much space is required to hold the data and additional database objects will not be enough. Each disk subsystem has its own throughput limitations, as do the Enterprise servers (hosts) to which they attach. And each application has its own concurrents or expectations about how available the data will be.

How do you know what throughput you will need? The best way (and not always an option) is to measure the performance of an existing system. If you do not have an existing system to measure, consider referencing benchmarks that have been completed for similar applications.

To complete an accurate sizing picture, it will be essential to know what the characteristics of your workload are. In Chapter 3, "Configuration for performance and manageability" on page 29 we discussed two main workload types: Online Transaction processing, and Decision Support Systems. From a disk subsystem perspective their characteristics can be summarized as follows:

OLTP

From a workload perspective, OLTP databases typically:

- Process a large number of concurrent user sessions
- Process a large number of transactions using simple SQL statements
- Process a single database row at a time
- Are expected to complete transactions in seconds, not minutes or hours

This equates to a large number of small block transactions to disk.

For the disk subsystem, this workload often appears as:

- Random access of 4K-8K records
- MIxture of reads and writes (70% reads is typical)

For this type of workload, pay attention to the number of I/O per second that the configuration can support. Smaller capacity disks may be most appropriate, especially if the workload has a significant amount of update activity.

DSS

DSS systems typically deal with substantially larger volumes of data than OLTP systems due to their role in supplying users with large amounts of historical data. Whereas 100 gigabytes would be considered large for an OLTP system, a large DSS system would most likely be 1 terabyte or more. The increased storage requirements of DSS systems can also be attributed to the fact that they often contain multiple, aggregrated views of the same data.

For the disk subsystem, the workloads often contain:

- Table scans, which result in sequential accesses of fairly large records (128KB)
- High read content

For this type of workload:

- Pay attention to the bandwidth of the subsystem as measured in MB/sec.
- Because of the sequential accesses, larger capacity disks may be most appropriate.

The following types of questions will also need to be considered:

- Is most of work that the database will be performing short, quick SQL statements?
- Do you have more long running queries?
- How many users will be connected to the system?
- Are there going to be availability requirements?
- Are there going to be requirements for user response times?
- What percentage of the work is read versus write activity?
- How random is the work?
- Will you be performing periodic bulk loads of the data?
- How does data enter the system?
- How is data removed from the system, or is history kept?

The most important measures to obtain from an existing disk subsystem include: the number of I/Os per second, the total MB/sec transferred, the KB/sec Read, and the KB/sec Written.

Why throughput?

Each of the questions above can effect the need to have more disks available to the system than a pure capacity analysis would indicate. For example, a single disk array can sustain a specific transfer rate for sequential write activity and we know that the data volume will fit easily on the disk array. We may still need to provide two disk arrays to the application if the amount of data to be loaded is required to be available to users before the time it would take to load onto one disk array. Adding the second disk array may cut the elapsed time (almost in half) and make the data available to users that much quicker.

This is where the information that you can gather from existing systems can be utilized. If you know the IO statistics from an existing application, you can estimate how long it would take to perform the same application using an ESS system.

Using benchmarking data

Each of the questions listed earlier in this section will effect the calculations made using benchmark data as well.

4.2 How to size the disk and CPU

Knowing how much storage space is required and what your throughput requirements and workload are like is a great start. From there you will need to understand the capabilities of the ESS system, as well as the host system to which it will be attached.

4.2.1 ESS throughput capabilities

Once you have determined what your throughput requirements are, you should determine how many of the various ESS components (disk arms, arrays, disk adapters, clusters) will be be required to achieve that rate. Throughput rates for the ESS can be found at:
`http://w3.rmss.storage.ibm.com/hardsoft/products/ess/whitepaper.htm`

Please refer to your IBM Storage System Specialist if you do not have access to this Web site.

Once you know the capabilities of the ESS in terms of throughput, you need to match this information with the host system(s) to which it will be connecting.

4.2.2 Host systems throughput capabilities

Each Enterprise Server (host system) has a published maximum I/O rate. It is based on many components of the system, such as how many disk adapters or Fibre channel adapters are allowed to attach.

Each host system has built into it the capability of attaching IO devices through Fiber channel cards and/or SCSI adapters. There will always be a maximum number of attachments that a particular system is capable of. These limitations are published in the announcement letters for the host system.

If you have determined that you will need more ESS subsystems than your chosen host system can support, you will need to add additional host systems and connect them through a SP switch or high-speed fiber interconnect.

Note: If more than one host system will be clustered together and you still want a single view of the database across these physical nodes, you must use DB2 UDB EEE.

5

Mapping ESS resources to AIX and DB2 UDB

This chapter describes how to map the logical devices from either the AIX perspective or the DB2 UDB perspective to the ESS perspective.

When working with any system it is essential to understand what resources are being used, where and when. This is true even when working with ESS which masks the activity of a single disk. Understanding how the database is laid out on the underlying hardware (for ESS at the array level) is necessary when performance issues arise or when considering how additional applications should be incorporated into the system.

Questions, such as these, can only be answered after an understanding of the layout of the system is achieved:

- What resources is my Database using?
- Is my I/O workload spread evenly?
- Do I have hotspots?
- Am I getting the most out of the hardware I have purchased?
- How many disks am I using per data partition?

In order to accomplish this, three “points of view” or perspectives are required: ESS, AIX, and DB2 UDB.

5.1 ESS perspective

The ESS resources can be identified down to the array level. StorWatch Specialist and StorWatch Expert can be used to capture and display configuration information as well as performance data.

5.1.1 StorWatch Specialist

The StorWatch Specialist Web-based interface will show the layout of the ESS in either a tabular format or a visual representation.

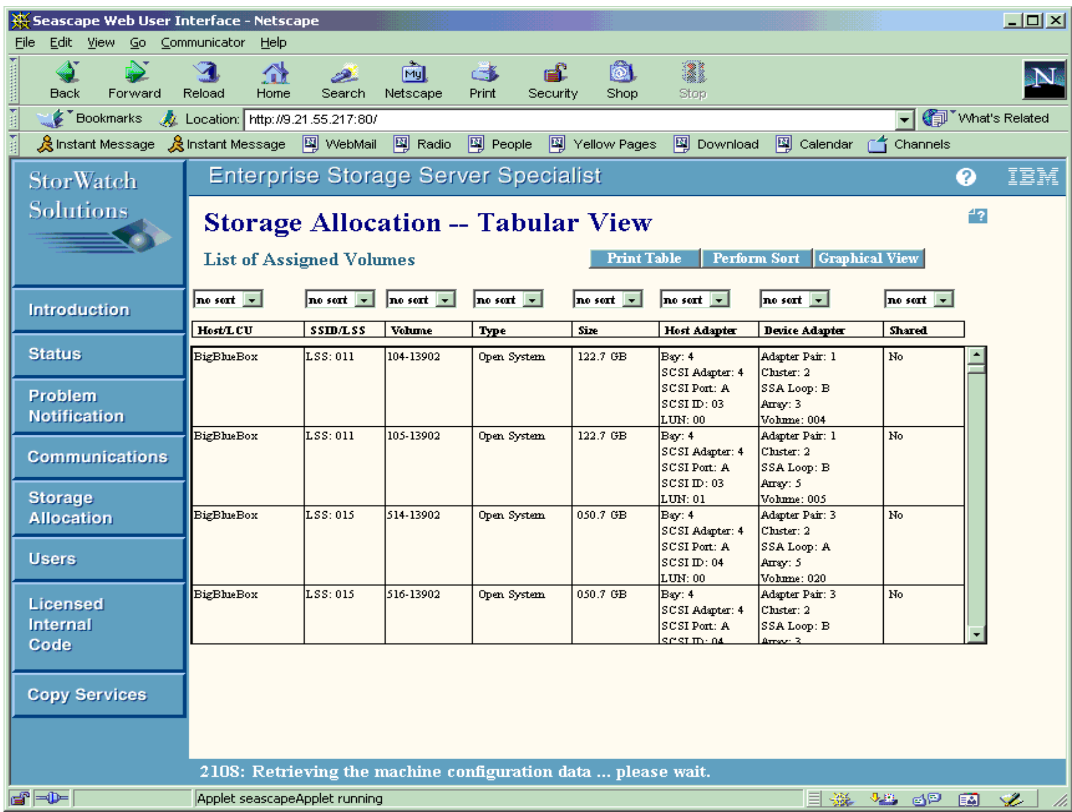


Figure 5-1 StorWatch Specialist tabular view

The tabular view (seen in Figure 5-1 on page 54) shows each ESS logical disk (sometimes referred to as a LUN) with pertinent information, such as how large the space is that has been set aside for this logical disk, what it's assigned volume is, and which host and device adapters it is connected to.

> **Note:** The VOLUME column contains the data which will allow us to map from AIX to the ESS and back.

There will be one row per logical disk and device connection. This listing can be saved as text and then browsed using any text editor. This information is also made available to the StorWatch Expert through the scheduling and execution of the StorWatch data collection task.

The graphical view of the ESS subsystem is extremely useful. It can be used to quickly and easily verify whether multiple paths have been defined for a subsystem, and to understand through which host adapters a particular array is connected.

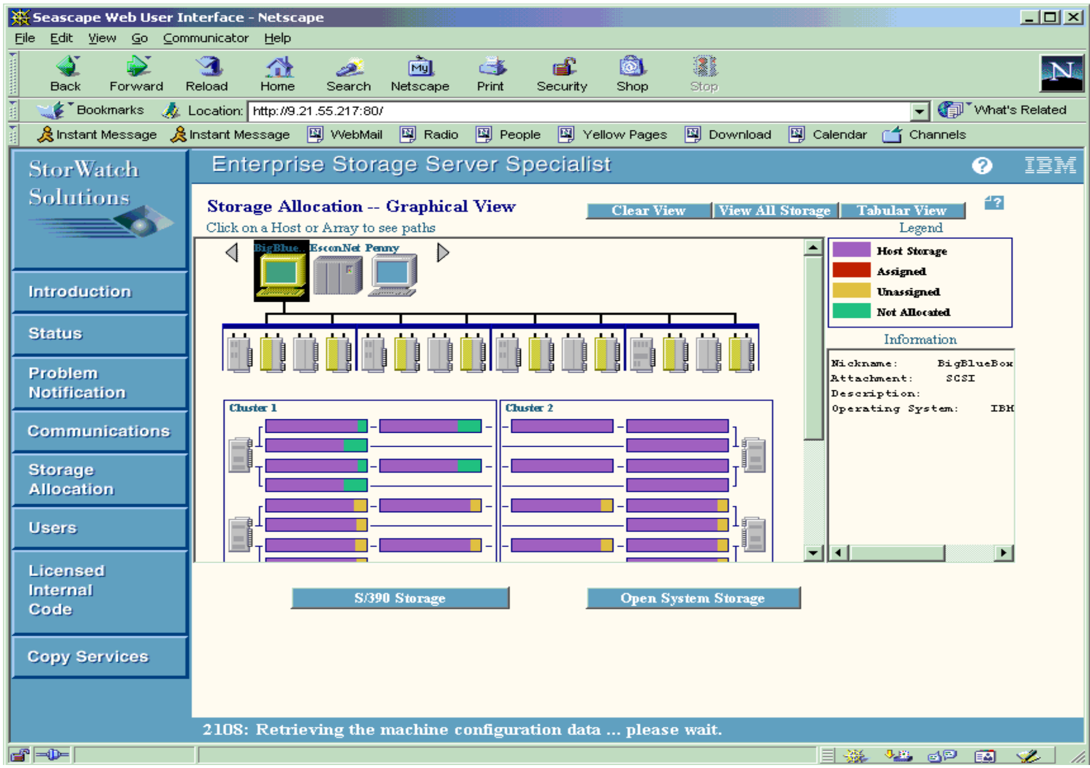


Figure 5-2 StorWatch Specialist graphical view

In addition to showing the contents of the arrays within the ESS, the visual representation can show the paths that are currently enabled for a particular array.

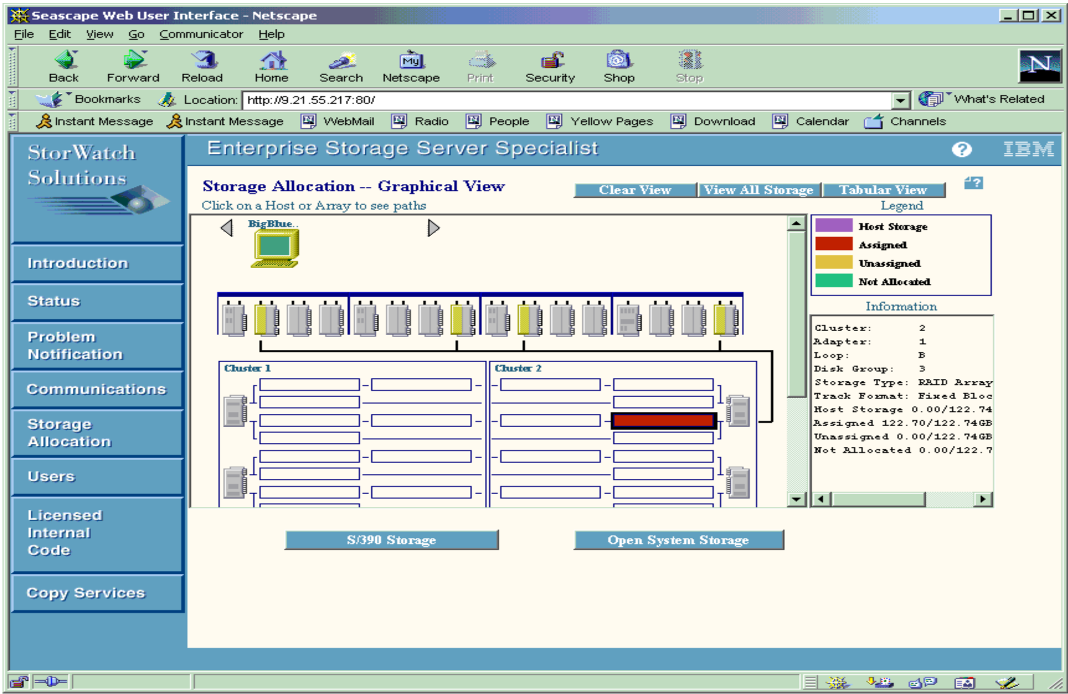


Figure 5-3 StorWatch Specialist view of paths to an array

As you can see in Figure 5-3 on page 56, the array highlighted has four paths defined to its host system.

5.1.2 StorWatch Expert

StorWatch Expert is used to gather performance data for the ESS. As such, we will discuss it in more detail in Chapter 6, “Diagnostics and performance monitoring” on page 69. However, StorWatch Expert contains hiperlinks to the StorWatch Specialist configuration views. It has the added advantage of being based on an underlying DB2 UDB relational database structure. This means that once you have populated the StorWatch Expert database you can then run your own queries against it.

5.2 AIX perspective

AIX resources can be viewed from a logical volume perspective and from a physical volume perspective. In order to map AIX resources to the underlying ESS, we will need to understand how both the physical volumes trace to ESS logical disks and how the AIX volume groups overlay these resources.

The AIX operating system and the optional (and recommended) Subsystem Device Driver (SDD) software provide multiple methods for obtaining configuration information. While SMITTY may be used to capture / define many of these resources, we will identify the underlying system commands that can be used to view the configuration information.

5.2.1 Virtual paths - Subsystem Device Driver (SDD)

The concept of a virtual path comes into play when more than one connection between the ESS and the AIX server are provided. Virtual paths (vpaths), allow the system to take advantage of the multiple host adapters, allowing greater overall I/O bandwidth.

From the AIX point of view, the resource that needs to be mapped (an hdisk from iostat), now translates first into a vpath and then through that vpath to the ESS volume. Figure 5-4 on page 57, shows the graphical view of the ESS. We have manually identified the hdisks and vpath that the resources map to.

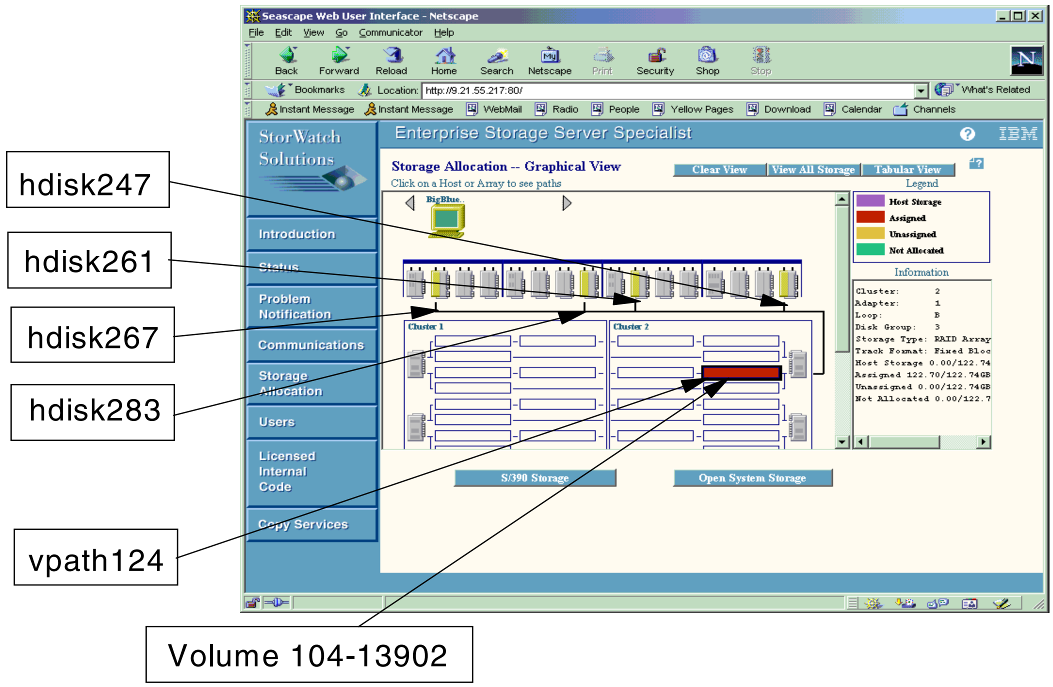


Figure 5-4 AIX resources mapped to ESS graphical layout

The commands outlined in the next section are necessary to be able to track the AIX perspective through to the DB2 UDB tablespace definitions. However, these commands alone will not provide enough information to track the disk activity (as reported by iostat) back to the ESS. In order to do that, we need the virtual path configuration.

Virtual path configuration — lsvpcfg

Similar to many of the AIX commands, the **lsvpcfg** command can be executed without parameters. In this case, all of the vpaths which are defined will be returned.

The output of the **lsvpcfg** command will tell us how many virtual paths are defined for a particular resource. This is important when reviewing I/O activity. In order to calculate the throughput for a particular disk group (array), all AIX physical volume (hdisk) activity for the same vpath should be added together. In the case where multiple ESS logical disks have been defined over a single array, multiple vpaths' statistics may need to be added together to obtain the throughput for the entire array.

In the example below, vpath0, vpath1, vpath3, vpath6, vpath7 and vpath9 have one single path defined, while vpath2, vpath4 and vpath5 have eight paths defined.

The information which precedes the equals sign is the volume number of the ESS logical disk associated with this vpath. To match this with the ESS Specialist displays, a hyphen needs to be inserted after the third digit. (For example 31213902 shown in Example 5-1 becomes 312-13902.)

Example 5-1 lsvpcfg command

```
vpath0 (Avail pv ssa4vg3) 31213902 = hdisk140 (Avail )
vpath1 (Avail pv ssa4vg3) 31313902 = hdisk141 (Avail )
vpath2 (Avail pv ssa4vg3) 31413902 = hdisk142 (Avail ) hdisk182 (Avail ) hdisk183 (Avail )
hdisk184 (Avail ) hdisk185 (Avail ) hdisk186 (Avail ) hdisk187 (Avail ) hdisk188 (Avail )
vpath3 (Avail pv ssa4vg3) 31513902 = hdisk143 (Avail )
vpath4 (Avail pv ssa4vg3) 31613902 = hdisk144 (Avail ) hdisk189 (Avail ) hdisk190 (Avail )
hdisk191 (Avail ) hdisk192 (Avail ) hdisk193 (Avail ) hdisk194 (Avail ) hdisk195 (Avail )
vpath5 (Avail pv ssa4vg3) 31713902 = hdisk145 (Avail ) hdisk196 (Avail ) hdisk197 (Avail )
hdisk198 (Avail ) hdisk199 (Avail ) hdisk200 (Avail ) hdisk201 (Avail ) hdisk202 (Avail )
vpath6 (Def pv ) 51213902 = hdisk176 (Avail pv ssa6vg3)
vpath7 (Def pv ) 51313902 = hdisk177 (Avail pv ssa6vg3)
vpath8 (Def pv ) 51413902 = hdisk178 (Avail pv ssa6vg3) hdisk203 (Avail pv ssa6vg3)
hdisk204 (Avail pv ssa6vg3) hdisk205 (Avail pv ssa6vg3) hdisk206 (Avail pv ssa6vg3)
hdisk207 (Avail pv ssa6vg3) hdisk208 (Avail pv ssa6vg3) hdisk209 (Avail pv ssa6vg3)
vpath9 (Def pv ) 51513902 = hdisk179 (Avail pv ssa6vg3)
```

In the above example, vpath8 refers to ESS Volume 514-13902. This can be mapped to adapter pair 3, Cluster 2, SSA Loop A, Array 5 of the ESS system (as shown in Figure 5-1 on page 54).

It is also important to review these definitions to understand whether all resources are assigned the same number of virtual paths. (In Example 5-1 on page 58, some had single paths, others have eight). When reviewing the iostat information, the rates for an ESS logical disk with a single path will look much higher than the rates for ESS logical disks with multiple paths. This will initially look as if the system is not balancing its workload correctly.

5.2.2 Logical Volume Manager

The Logical Volume Manager is used to make physical disks (in the case of ESS these are logical disks) available for use by applications. Physical disks can be combined into a single volume group, and volume groups can be divided into multiple logical volumes. In order to determine where these resources really reside on an ESS, information from the following commands is necessary:

List volume group — lsvg

The `lsvg` command will show how large the volume group is in terms of physical partition sizes. It also shows status information about the volume group.

Example 5-2 lsvg command

```
VOLUME GROUP:   ssa6vg3                  VG IDENTIFIER:  000b03bd73262d81
VG STATE:       active                   PP SIZE:        256 megabyte(s)
VG PERMISSION:  read/write               TOTAL PPs:      998 (255488 megabytes)
MAX LVs:        256                      FREE PPs:       37 (9472 megabytes)
LVs:            13                       USED PPs:       961 (246016 megabytes)
OPEN LVs:       5                        QUORUM:         4
TOTAL PVs:      6                        VG DESCRIPTORS: 6
STALE PVs:      0                        STALE PPs:      0
ACTIVE PVs:     6                        AUTO ON:        yes
MAX PPs per PV: 1016                     MAX PVs:        32
```

With the -p option, the **lsvg** command will list all of the physical volumes which have been configured within the volume group. In the case outlined in the example below, the ssa6vg3 volume group has been defined across six ESS logical disks, shown here as vpaths (which indicates that multi-pathing has been enabled). If no multi-pathing has been enabled, the physical volumes names shown here would normally begin with ‘hdisk’.

Example 5-3 lsvg -p command

```
ssa6vg3:
PV_NAME          PV STATE    TOTAL PPs   FREE PPs    FREE DISTRIBUTION
vpath6           active      123         3           00..00..00..00..03
vpath7           active      188         7           00..00..00..00..07
vpath8           active      188         8           00..00..00..00..08
vpath9           active      123         3           00..00..00..00..03
vpath10          active      188         8           00..00..00..00..08
vpath11          active      188         8           00..00..00..00..08
```

Using the -l option with the **lsvg** command will display the logical volumes associated with this volume group. This information is required when mapping these AIX resources to the DB2 tablespaces.

In the example below, you can see that the logical volumes were named with the original hdisk identifier. When using multi-pathing, the hdisks associated with a particular ESS logical disk can change when new vpaths are introduced. As such, it is not advisable to use the hdisk nomenclature within your logical volume names. Other naming conventions can be used to indicate where the logical volume resides within the ESS, such as ‘da3a1’ to indicate disk adapter 3 and array 1, these logical naming conventions won’t be affected by re-configuring the ESS paths available to a logical disk.

Example 5-4 lsvg -l command

```
ssa6vg3:
LV NAME            TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
lv2a.kwai.hd165    raw        90    90    1    closed/syncd  N/A
lv2b.kwai.hd165    raw        90    90    1    closed/syncd  N/A
lv2a.kwai.vp8      jfs        90    90    1    open/syncd    /fs2a.kwai.vp8
lv2b.kwai.vp8      jfs        90    90    1    open/syncd    /fs2b.kwai.vp8
lv2a.kwai.vp10     jfs        90    90    1    open/syncd    /fs2a.kwai.vp10
lv2b.kwai.vp10     jfs        90    90    1    open/syncd    /fs2b.kwai.vp10
lv2a.kwai.hd169    raw        90    90    1    closed/syncd  N/A
lv2b.kwai.hd169    raw        90    90    1    closed/syncd  N/A
lv2a.kwai.hd164    raw        60    60    1    closed/syncd  N/A
lv2b.kwai.hd164    raw        60    60    1    closed/syncd  N/A
lv2a.kwai.hd167    raw        60    60    1    closed/syncd  N/A
lv2b.kwai.hd167    raw        60    60    1    closed/syncd  N/A
loglv23            jfslog     1     1     1    open/syncd    N/A
```

List physical volume — lspv

Since the volume group can cover multiple ESS logical disks, we need to use the List physical volumes command to determine where a particular logical volume resides. In the example shown below, twelve logical volumes have been defined on the ‘vpath30’ physical volume.

Example 5-5 lspv -l command

```
vpath30:
LV NAME              LPs   PPs   DISTRIBUTION          MOUNT POINT
lv.rb.D1.DA3a5       5     5     05..00..00..00..00    N/A
lv.rb.D1.DA3a4       5     5     05..00..00..00..00    N/A
lv.rb.T1.DA3a4       5     5     05..00..00..00..00    /fs.rb.T1.DA3a4
lv.rb.D1.DA3a3       5     5     05..00..00..00..00    N/A
```

```
lv.rb.T1.DA3a3        5     5     05..00..00..00..00    /fs.rb.T1.DA3a3
lv.rb.T1.DA3a1        5     5     00..05..00..00..00    /fs.rb.T1.DA3a1
lv.rb.D1.DA3a1        5     5     00..05..00..00..00    N/A
lv.rb.T1.DA3a2        5     5     00..05..00..00..00    /fs.rb.T1.DA3a2
lv.rb.D1.DA3a2        5     5     00..05..00..00..00    N/A
lv.rb.T1.DA3a5        5     5     00..04..01..00..00    /fs.rb.T1.DA3a5
lv.rb.T1.DA3a6        5     5     00..00..05..00..00    /fs.rb.T1.DA3a6
lv.rb.D1.DA3a6        5     5     00..00..05..00..00    N/A
```

The following example shows the lspv command with the -p option. This gives us information about where from an AIX perspective the logical volumes reside. Mount points for the logical file systems are noted, for raw devices, the mount point is N/A.

Example 5-6 lspv -p command

```
vpath30:
PP RANGE  STATE   REGION        LV NAME             TYPE       MOUNT POINT
  1-5     used    outer edge    lv.rb.D1.DA3a5      raw        N/A
  6-10    used    outer edge    lv.rb.D1.DA3a4      raw        N/A
 11-15    used    outer edge    lv.rb.T1.DA3a4      jfs        /fs.rb.T1.DA3a4
 16-20    used    outer edge    lv.rb.D1.DA3a3      raw        N/A
 21-25    used    outer edge    lv.rb.T1.DA3a3      jfs        /fs.rb.T1.DA3a3
 26-30    used    outer middle  lv.rb.T1.DA3a1      jfs        /fs.rb.T1.DA3a1
 31-35    used    outer middle  lv.rb.D1.DA3a1      raw        N/A
 36-40    used    outer middle  lv.rb.T1.DA3a2      jfs        /fs.rb.T1.DA3a2
 41-45    used    outer middle  lv.rb.D1.DA3a2      raw        N/A
 46-49    used    outer middle  lv.rb.T1.DA3a5      jfs        /fs.rb.T1.DA3a5
 50-50    used    center        lv.rb.T1.DA3a5      jfs        /fs.rb.T1.DA3a5
 51-55    used    center        lv.rb.T1.DA3a6      jfs        /fs.rb.T1.DA3a6
 56-60    used    center        lv.rb.D1.DA3a6      raw        N/A
 61-73    free    center
 74-97    free    inner middle
 98-122   free    inner edge
```

List logical volume — lslv

The list logical volume command allows you to identify the physical volume that a logical volume resides upon. This is especially useful when reviewing (manually) where logical volumes that have been defined to DB2 UDB reside on the ESS.

Example 5-7 lslv -l command

```
lv.rb.T1.DA3a1:/fs.rb.T1.DA3a1
PV                COPIES        IN BAND       DISTRIBUTION
vpath96           005:000:000   100%          000:005:000:000:000
```

5.3 DB2 UDB perspective

DB2 UDB uses a tablespace and container concept to provide a mapping of its relational structures to the operating system resources which have been made available to it. A description of these database objects can be found in Chapter 2, “Terminology and concepts” on page 13.

5.3.1 List tablespaces

The **list tablespace** command is used to obtain information about where existing tablespaces reside on disk. There are multiple forms of this command which return varying levels of detail.

The base level of this command returns a list of tablespaces that have been defined to the system. In the following Example 5-8, we are showing the results for only one tablespace. Its name is TESTDDATA and its Tablespace ID is 4.

Example 5-8 List tablespace command

```
db2 list tablespaces show detail

Tablespace ID                        = 4
 Name                                 = TESTDATA
 Type                                 = Database managed space
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     Normal
 Total pages                          = 2949120
 Useable pages                        = 2948544
 Used pages                           = 1253040
 Free pages                           = 1695504
 High water mark (pages)              = 1253072
 Page size (bytes)                    = 16384
 Extent size (pages)                  = 16
 Prefetch size (pages)                = 96
 Number of containers                 = 36
for the data tablespace.
```

The next level of detail adds the container clause and shows the logical location of the container in the disk subsystem. In order to run this command, you will need to know the tablespace ID that you want to map. This can be seen in Example 5-9.

Example 5-9 List tablespace containers command

```
db2 list tablespace containers for 4

            Tablespace Containers for Tablespace 4

 Container ID                         = 0
 Name                                 = /dev/rlv.rb.D1.DA3a2
 Type                                 = Disk

 Container ID                         = 1
 Name                                 = /dev/rlv.rb.D1.DA3b2
 Type                                 = Disk

 Container ID                         = 2
 Name                                 = /dev/rlv.rb.D1.DA3c2
 Type                                 = Disk

(partial listing)
```

As you can see in Example 5-9, the command shows the underlying AIX resource which DB2 UDB will utilize when storing database objects for this tablespace.

In a partitioned database environment (EEE), only tablespaces that are defined to the node you are connected to will show. You can use the 'rah' capabilities of AIX to issue commands on the remote servers. For more information on the use of **rah** and **db2_all** please refer to Appendix D of the *DB2 UDB Administration Guide: Performance V7*, SC09-2945.

Example 5-9 was an example of a DMS raw table space and below (Example 5-10) is an example of an SMS table space.

Example 5-10 SMS containers

```
list tablespace containers for 3

            Tablespace Containers for Tablespace 3

 Container ID                         = 0
 Name                                 = /fs.rb.T1.DA3a1/reg64/node0
 Type                                 = Path

 Container ID                         = 1
 Name                                 = /fs.rb.T1.DA3b1/reg64/node0
 Type                                 = Path

 Container ID                         = 2
 Name                                 = /fs.rb.T1.DA3c1/reg64/node0
 Type                                 = Path

(partial listing)
```

5.3.2 Log files and directories

DB2 UDB has a very robust backup and recovery method. Recovery is based on transaction logging. The information about where the transaction log files are located on disk is maintained in the database configuration file.

To find out where your log files are stored you can either issue the following command, or view the database configuration file from the Control Center.

Example 5-11 List database configuration for log file location

```
db2 list db cfg for TEST | grep "log file"

Number of primary log files              (LOGPRIMARY) = 8
Number of secondary log files             (LOGSECOND) = 2
Changed path to log files                (NEWLOGPATH) =
Path to log files                                     = /kwaitemp/LOGS/reg64/NODE0000/
First active log file                                 =
```

5.4 Bringing the various components together

In this section we detail how the various components that are used can be brought together so that an overall view can be obtained.

5.4.1 Externalizing StorWatch configuration data

Having the ESS configuration data available to you from within StorWatch Specialist and StorWatch Expert is great. However, you will need to combine this information with data that is not accessible to StorWatch at this time—AIX and DB2 UDB data. In order to relate the ESS to AIX and/or DB2 UDB, you need to obtain the VOLUME number from the AIX configuration as well as the hdisk/vpath that you are trying to identify.

A manual approach

The first method is a manual approach. You simply note down the resource you are trying to identify/map (for example, a particular hdisk/volume/vdisk), and search the online screens for a match. This method works great, but can be rather slow.

Text-based approach

This method requires that you gather the configuration information from the StorWatch Specialist using the Web-based interface. You then request to print the results of the tabular display. StorWatch Specialist then opens a second window, and the information contained can be printed to a file, or saved in Html or other format.

Once a flat file is obtained, you can use familiar searching mechanisms to identify/map resources (for example, 'find' in a text editor, or 'grep' from a UNIX system).

You may find this information even more accessible, if you further process the output of the tabular display creating a single line per logical disk with the cluster, adapter, loop and array/disk group information identified. This method may work faster for you, especially if you are used to a UNIX command line interface. However, it does require some semi-manual steps (you can always create a script to help automate it), and you need to ensure (manually) that you note the date on which you captured your configuration information. This is especially important as one of the features of the ESS is its ease of re-configuration and you should not assume that it will stay unchanged.

Relational approach

Since the configuration data can be stored in a DB2 table (if you have ESS Expert), you have the option of accessing the table itself. This can be done through an online query of the ESS Expert database, or you can export the appropriate configuration tables (vvolx is the main one), and import them into your own performance/configuration/mapping database.

> **Note:** You will need to ensure that you have access to the ESS Expert database with read privileges.

If you are comfortable with SQL this can be a great approach. It has the added advantage of keeping in synch with the StorWatch Expert data capture task. (A configuration date is included in the vvolx table definition). The downside to this approach is that in order to take full advantage of it, you should load other information (for example, AIX volumes, and so on) into a database as well. This can take more preparation time (especially if loading iostat data).

During our work on this redbook, we populated the StorWatch Expert database tables and then exported the vvolx table information for use with our own mapping data.

5.4.2 Externalizing AIX, SDD and DB2 UDB configuration data

The commands we have shown you for the AIX, SDD, and DB2 UDB configurations readily produce information in the form of text files. We would recommend that you run these commands as your disk subsystem is modified. Keeping track of when the configuration data was captured is also beneficial so that it can be mapped to its matching ESS configuration data.

A manual approach

As in the ESS configuration data, you may simply run the AIX, SDD and DB2 UDB commands for a single resource which is of interest.

Text-based approach

The method requires that you gather the information from the previous commands and keep the results in text files on your system. Issuing the command and re-directing the output to a file will do the trick.

Again, once the configuration data is captured in files, you can use familiar searching mechanisms to identify / map resources (for example, ‘find’ in a text editor, or ‘grep’ from a UNIX system).

Relational approach

Once the text-based files have been captured, you can prepare them to be loaded into a relational database.

During the work on this redbook, we populated one DB2 UDB table with information from the **`lsvpcfg`** command and another with information from the **lspv** command. The tables could then be queried to help analyze I/O statistics that are captured during database processing.

5.4.3 Sample ESS layout

The following example worksheets can be used to help visualize your current or planned ESS system. They are designed to show how your AIX file systems or your DB2 UDB databases are laid out on top of the ESS structure. They can also be used to help picture performance activity on your system when combined with iostat data or StorWatch expert performance reports.

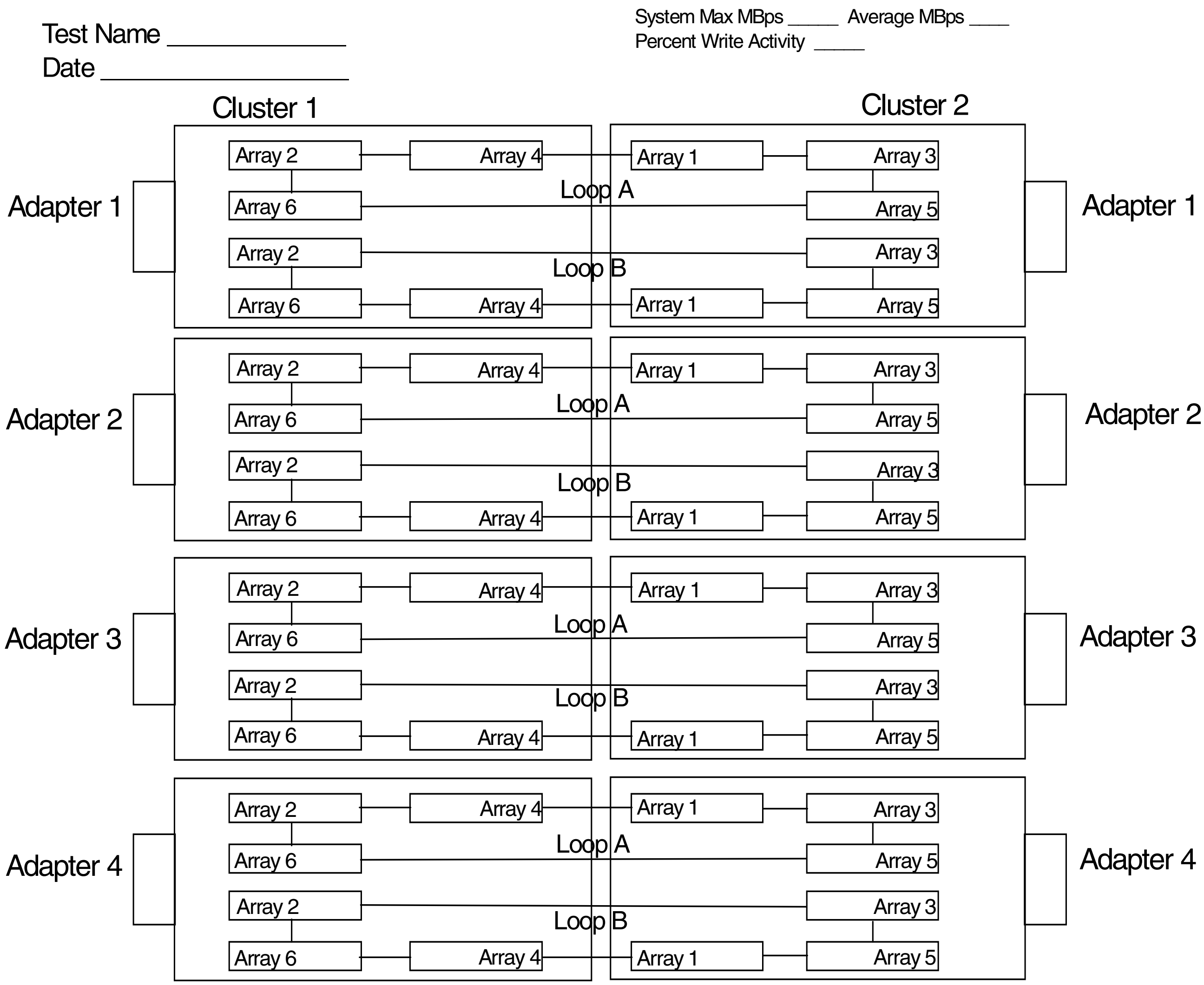


Figure 5-5 ESS layout worksheet

5.4.4 A DMS containers example

Lets use as an example the three DMS containers shown in Example 5-9 on page 61. Where do these reside on the ESS subsystem?

From the DB2 UDB configuration data, we see that the containers are associated with the following raw devices:

- /dev/rlv.rb.D1.DA3a2
- /dev/rlv.rb.D1.DA3b2
- /dev/rlv.rb.D1.DA3c2

There are a number of commands that we can use to find out where these devices are from the AIX point of view. In order to use this configuration data from DB2 for AIX, however, we need to remove the ‘/dev/r’ from the container definitions. Using the remaining device name, we can issue the command. As Example 5-12 shows, the containers in question reside on

vpath30, vpath31 and vpath32. We can now match these vpaths to their AIX physical volumes (hdisks) and to the ESS logical disk using the lsvpcfg command. The results of the lsvpcfg command can be reviewed to see how many paths (could be one) are defined for each.

Example 5-12 Locating the AIX physical volume for a DB2 container

```
lslv -l lv.rb.D1.DA3a2

PV                COPIES        IN BAND       DISTRIBUTION
vpath30           005:000:000   100%          000:005:000:000:000

lslv -l lv.rb.D1.DA3b2

PV                COPIES        IN BAND       DISTRIBUTION
vpath31           005:000:000   100%          000:005:000:000:000
lslv -l lv.rb.D1.DA3c2

PV                COPIES        IN BAND       DISTRIBUTION
vpath32           005:000:000   100%          000:005:000:000:000
```

.As Example 5-13 shows, each of the AIX physical volumes in question has eight virtual paths defined. The underlying ESS logical disks are identified as follows: 212-13902, 213-13902, and 214-13902.

Example 5-13 Identifying the physical volumes associated with a Virtual Path

```
lsvpcfg > lsvpcfg.out

grep vpath30 lsvpcfg.out
vpath30 (Avail pv ssa3vg3) 21213902 = hdisk72 (Avail ) hdisk297 (Avail ) hdisk303 (Avail )
hdisk309 (Avail ) hdisk315 (Avail ) hdisk321 (Avail ) hdisk327 (Avail ) hdisk333 (Avail )

grep vpath31 lsvpcfg.out
vpath31 (Avail pv ssa3vg3) 21313902 = hdisk73 (Avail ) hdisk298 (Avail ) hdisk304 (Avail )
hdisk310 (Avail ) hdisk316 (Avail ) hdisk322 (Avail ) hdisk328 (Avail ) hdisk334 (Avail )

grep vpath32 lsvpcfg.out
vpath32 (Avail pv ssa3vg3) 21413902 = hdisk74 (Avail ) hdisk299 (Avail ) hdisk305 (Avail )
hdisk311 (Avail ) hdisk317 (Avail ) hdisk323 (Avail ) hdisk329 (Avail ) hdisk335 (Avail )
```

The next step is to refer to the ESS Specialist configuration data, which will tell us which ESS resources are in use: cluster, disk adapter, loop, and array. This can be done by bringing up the configuration data (previously saved as an html document) or by using the StorWatch Specialist interface.

Once the tabular view is displayed, use your browsers search facility to locate the desired logical disk.

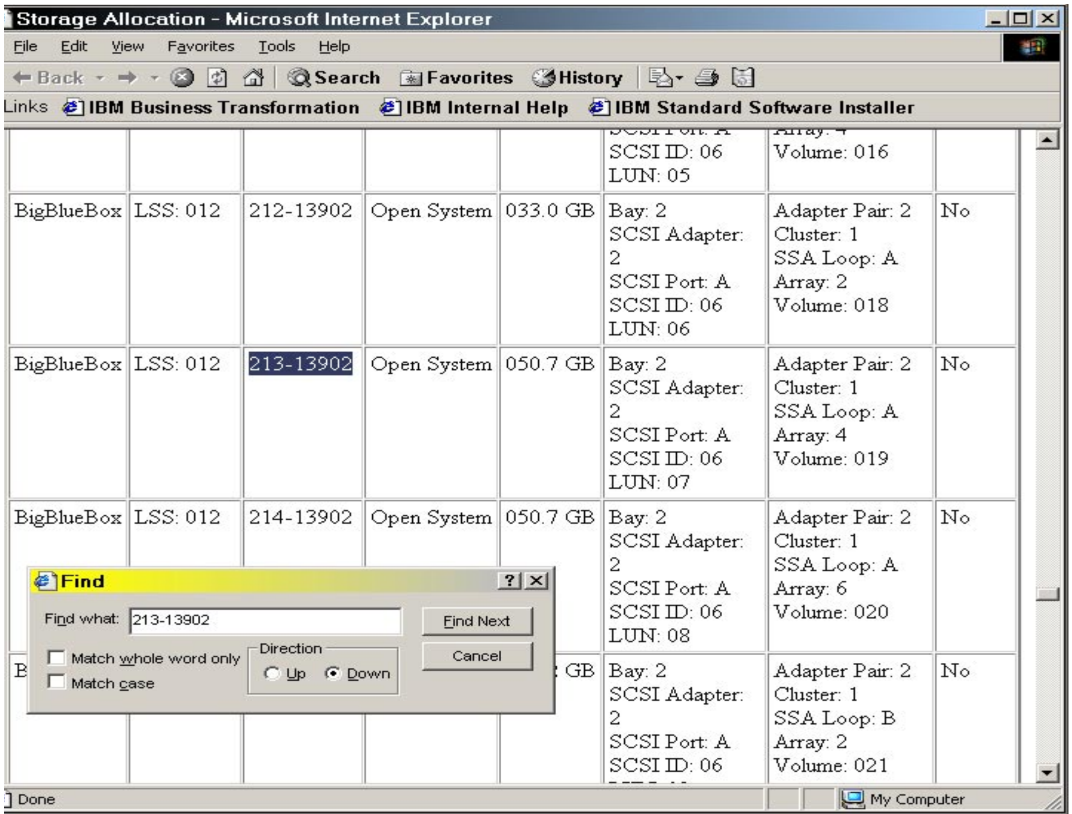


Figure 5-6 Searching the StorWatch Specialist tabular view

Once you have identified where the resource resides, you can use the sample ESS layout sheet to document this. This can be seen in Figure 5-7.

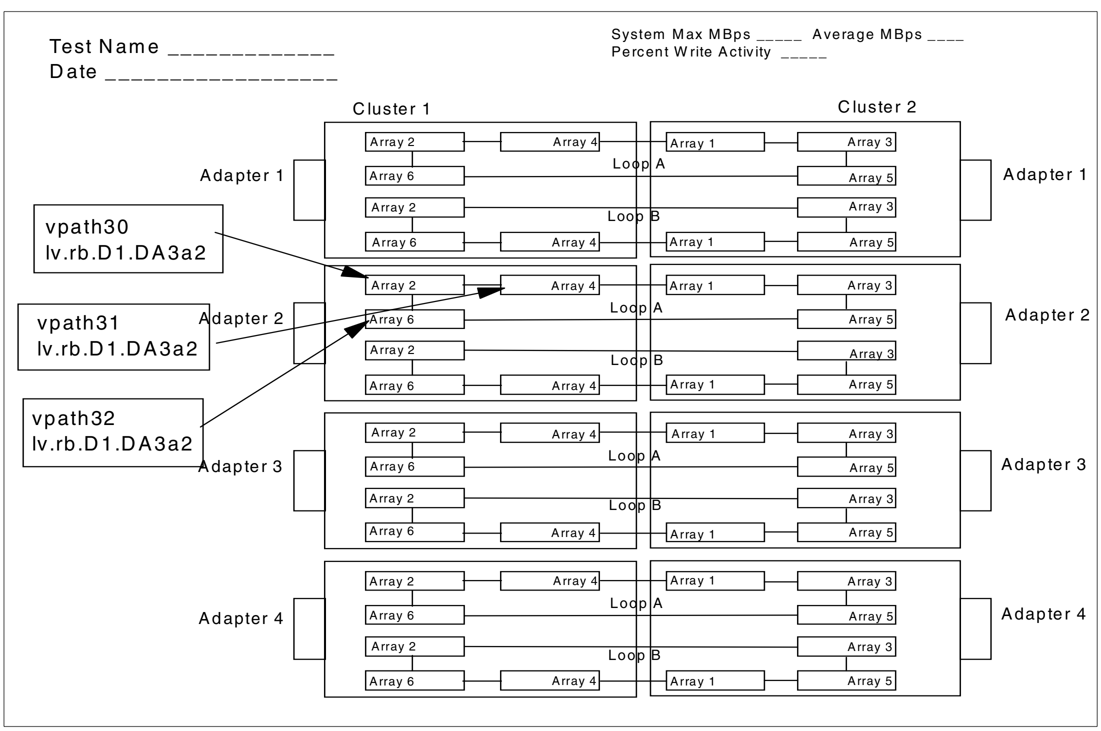


Figure 5-7 Using the worksheet to document physical volume resources

5.5 Using a configuration/mapping database

Once you have established a process for documenting the configuration data from the ESS, AIX and DB2 UDB systems, you can then make use of the combined information to provide a more tailored view of your system and how you use it.

For example, the configuration data can be joined with iostat information to provide a view of not only which I/O adapters are being used, but rates of data transfer they are achieving. The data can also be used to show which volume group, data partition or tablespace is providing that workload.

Running queries against this database will allow you to track subsets of I/O activity for the entire system (say for a particular performance test). And to summarize by cluster, adapter, loop or array. Taking additional steps to move this data into a end-user tool which provides charting capabilities, will allow you to provide a visual overview of how the system is performing and whether or not the workload is balanced across the various hardware resources: clusters, adapters, loops, and arrays as well as across the logical resources: data partitions.

This is the method that we used during the redbook project to produce the charts shown in Appendix A, “Data placement tests” on page 107.

5.6 Suggested outputs — preparing a map from your database

Here’s a list of outputs we suggest you keep up-to-date in an accessible location in the event of difficulties with your system. These will serve as a useful reference when you need to map from your DB2 database, through AIX and the AIX logical volume manager, the SDD software, and the ESS support software.

Table 5-1 Suggested outputs

Source	Command	Description
DB2	DB2 list tablespace containers for all tablespaces	Provides a listing of the OS device name used as a DB2 container.
AIX	mount or cat /etc/filesystems	Provides a listing of mounted devices.
AIX	ls /dev	Provides a listing of all devices, mounted or not.
ESS Specialist	tabular view of storage, saved as a .txt file	Provides a listing of each ESS logical volume, it’s identification, size, and adapter connections.
SDD (AIX command line)	lsvpcfg	Maps AIX hdisks to their virtual paths.
ESSutil (if you can get it)	lsess	Maps AIX hdisks to ESS logical volumes.

6

Diagnostics and performance monitoring

There are many components of an Enterprise Database Server: the host computer, the operating system, logical volume management software, multipathing software, I/O interconnects (possibly Fibre SAN), network hardware and software, DB2, the ESS management and analysis software, and of course the ESS itself. Unfortunately, there is no single tool you can use to track storage and diagnose problems across all of these components.

In this chapter we discuss some of the tools available, and show you how and when they can or should be used.

6.1 ESS performance and diagnostic tools

There are two main tools that are used to track performance for the ESS: IBM TotalStorage ESS Specialist and IBM Total Storage ESS Expert.

6.1.1 IBM TotalStorage ESS Specialist

The ESS Specialist Web-based tool comes with every ESS, and is used to configure storage allocation and monitor and report ESS status. Because it is a Web-based tool, the graphical interface of Specialist provides several helpful views of ESS status and configuration. Navigation is generally intuitive, but you should read and understand the instructions before using the buttons.

We have provided examples of how we use ESS Specialist in Chapter 5, “Mapping ESS resources to AIX and DB2 UDB” on page 53.

6.1.2 IBM TotalStorage ESS Expert

The ESS Expert software package, 5648-SWV, is a Web-based extension of ESS Specialist. Although it is a separate software package, it contains hyperlinks to the ESS Specialist configuration views. It makes use of software that runs internally to the ESS and collects statistics within the ESS itself. Several standard reports are available, and other custom reports can be created as well.

The redbook *IBM StorWatch Expert Hands-On Usage Guide*, SG24-6102, explains in detail how to set up and use this tool. This book can be found at `http://www.ibm.com/redbooks`. We have included here some examples of using this tool that are specifically related to performance.

Once the ESS Expert has been set up, the gathering of performance data must be scheduled. This is done through the Data Collection tasks. Figure 4-1, below, shows the data collection task setup screen. Using this screen you identify when and how long you want ESS Expert to gather information, and for which ESS you want statistics gathered.

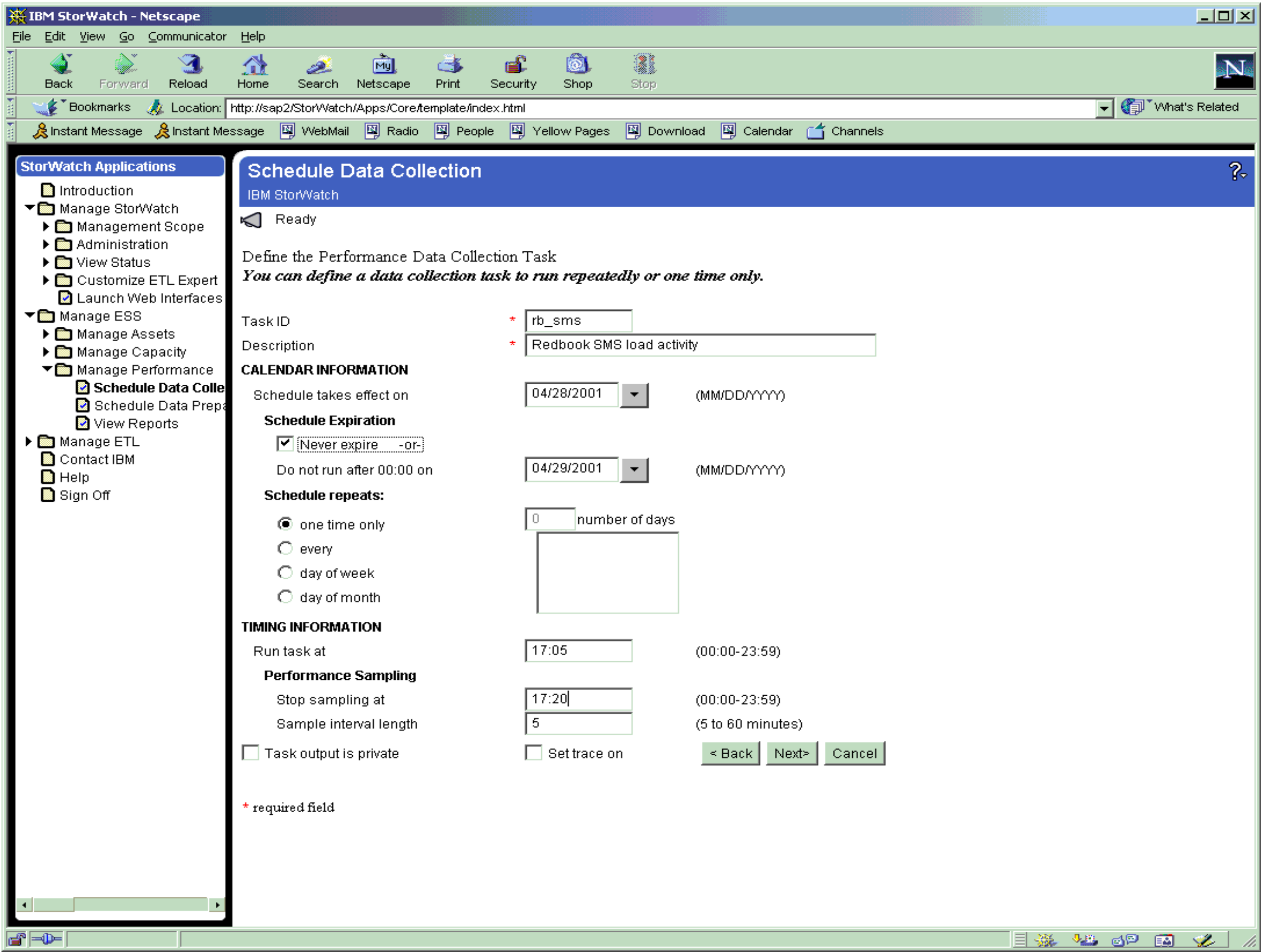


Figure 6-1 ESS Expert data collection setup

One of the items to note here, is that ESS Expert only allows us to capture data down to a minimum of a five minute interval. If you want a more granular view of the I/O activity (for example, down to the minute or second) you will need to use the AIX performance tools, which are discussed in the next section.

Once the data has been gathered, a number of ESS Expert reports are available. The following types of pre-defined reports can be built:

- Disk Utilization
- Disk Cache
- Cache Report

Each of those reports can be summarized at various levels of detail. Following are the four levels (the Disk Utilization is not available at the lowest or highest level):

- Cluster
- Disk Adapter
- Disk group (we have been referring to this as the array)
- Logical Volume (we have been referring to this as the ESS Logical disk or LUN)

ESS Expert — Disk Utilization reports

The following example, we show the summary of Disk Utilization reports. Selecting a particular date and Disk group will take you to the more detailed reports. The disk utilization reports show how busy the disk drive modules (DDMs) on an ESS were averaged over the time interval selected. The following example Figure 6-2 shows the type of report that be created.

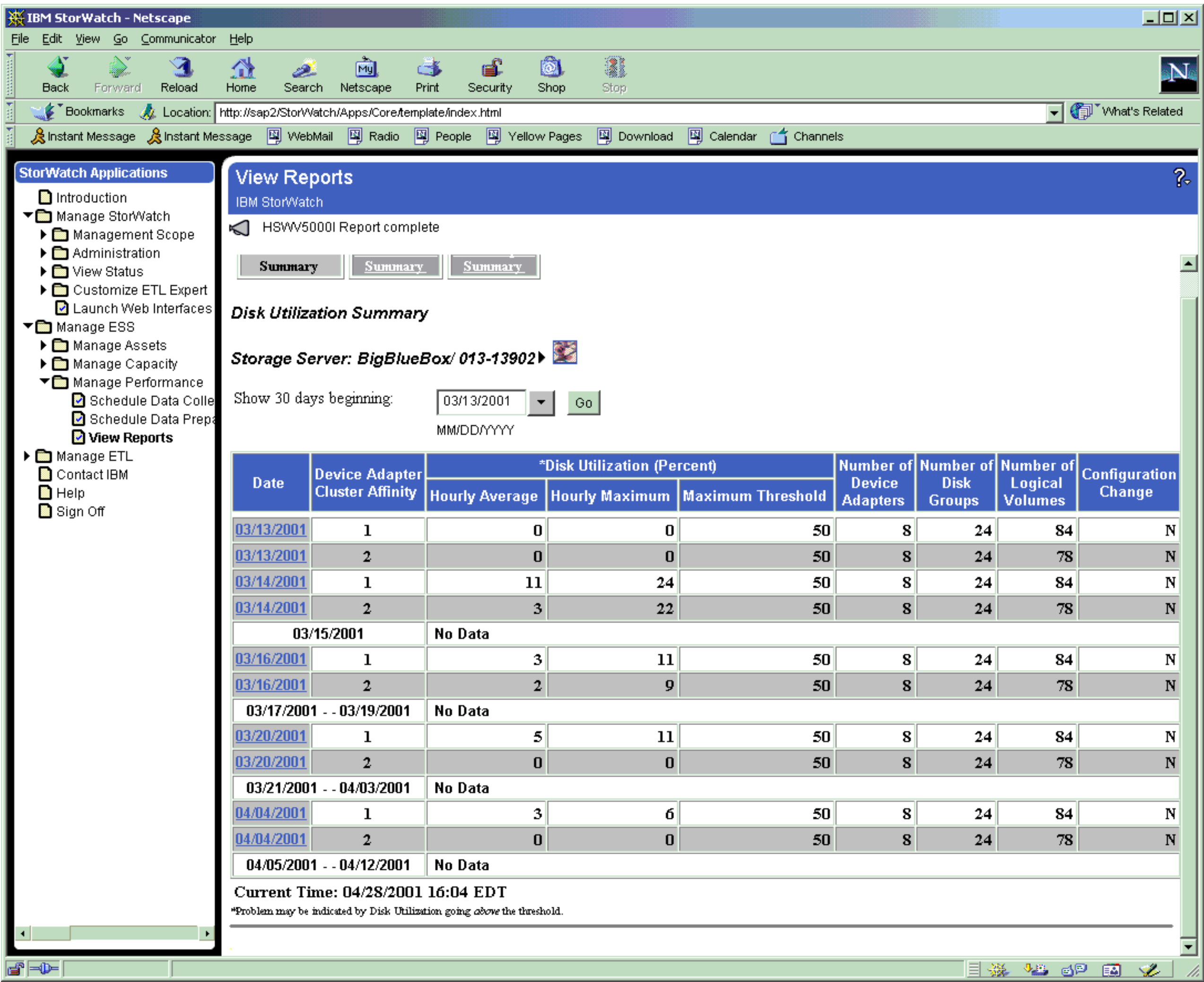


Figure 6-2 ESS Expert summary of disk utilization reports

The following examples, Figure 6-3 and Figure 6-4, show data collected by disk group. The first allows access to the second more detailed view over the time period collected. It also allows access to a graphical view of the data.

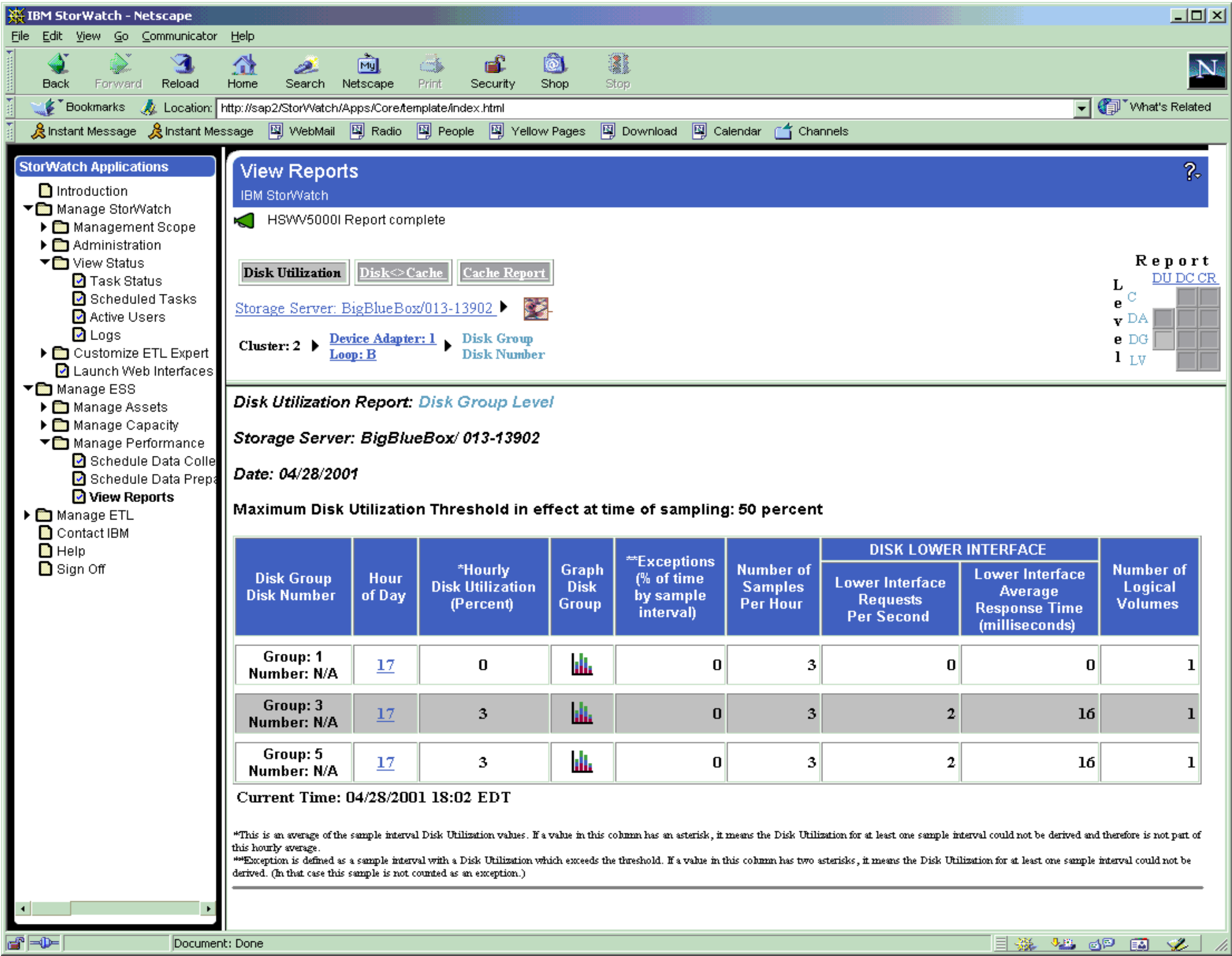
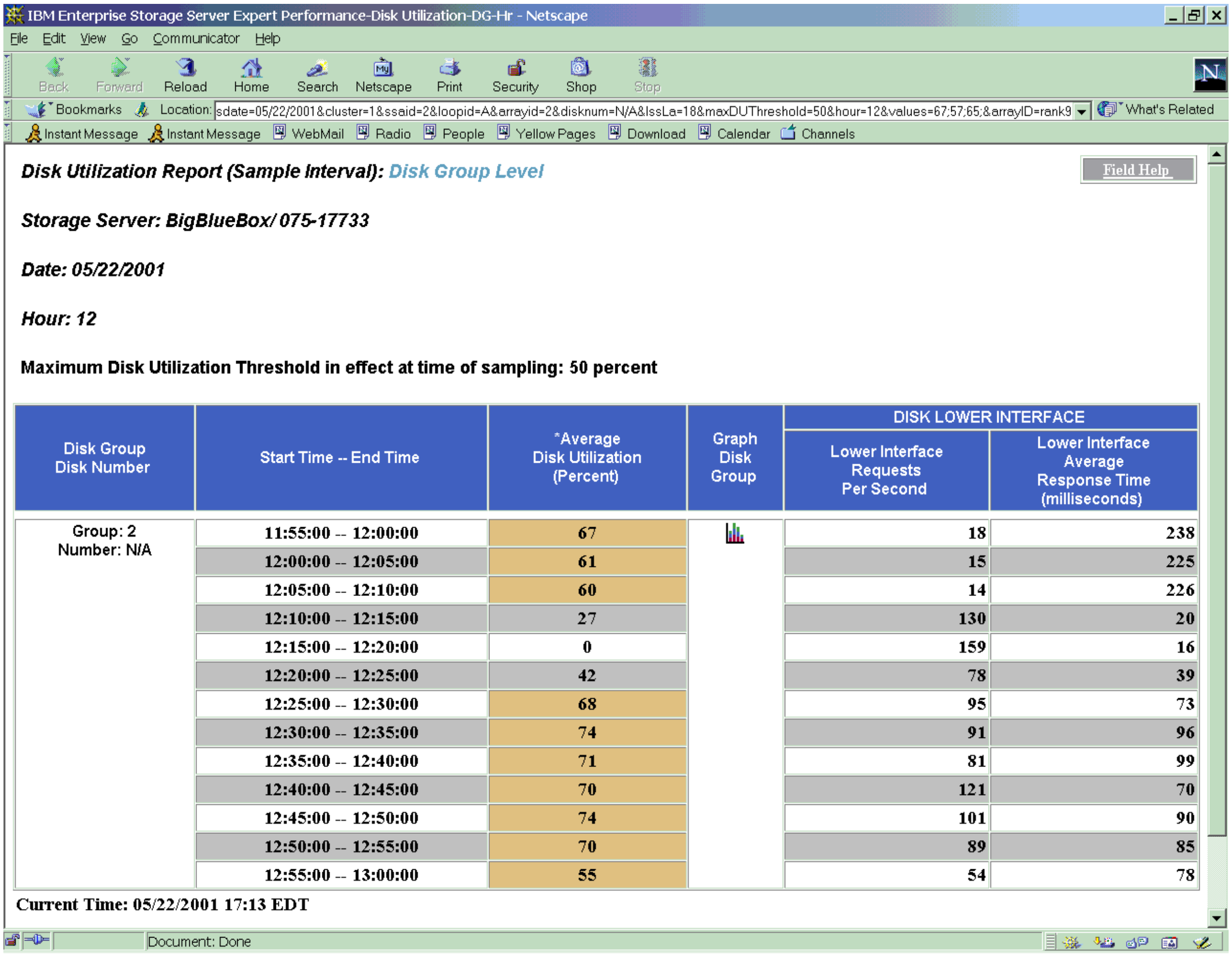


Figure 6-3 ESS Expert disk utilization report - disk group level, summary

The following example, Figure 6-3 shows data collection by disk group.



Disk Utilization Report (Sample Interval): Disk Group Level

Storage Server: BigBlueBox/ 075-17733

Date: 05/22/2001

Hour: 12

Maximum Disk Utilization Threshold in effect at time of sampling: 50 percent

Disk Group Disk Number	Start Time -- End Time	*Average Disk Utilization (Percent)	Graph Disk Group	DISK LOWER INTERFACE	
				Lower Interface Requests Per Second	Lower Interface Average Response Time (milliseconds)
Group: 2 Number: N/A	11:55:00 -- 12:00:00	67		18	238
	12:00:00 -- 12:05:00	61		15	225
	12:05:00 -- 12:10:00	60		14	226
	12:10:00 -- 12:15:00	27		130	20
	12:15:00 -- 12:20:00	0		159	16
	12:20:00 -- 12:25:00	42		78	39
	12:25:00 -- 12:30:00	68		95	73
	12:30:00 -- 12:35:00	74		91	96
	12:35:00 -- 12:40:00	71		81	99
	12:40:00 -- 12:45:00	70		121	70
	12:45:00 -- 12:50:00	74		101	90
	12:50:00 -- 12:55:00	70		89	85
	12:55:00 -- 13:00:00	55		54	78

Current Time: 05/22/2001 17:13 EDT

Figure 6-4 ESS Expert disk utilization report - disk group level, detail

ESS Expert — Disk to/from Cache reports

The Disk to/from Cache report shows you how many I/O operations occurred between the DDMs (disk device managers) and Cluster cache, known as staging and destaging. A typical report can be seen in Figure 6-5.

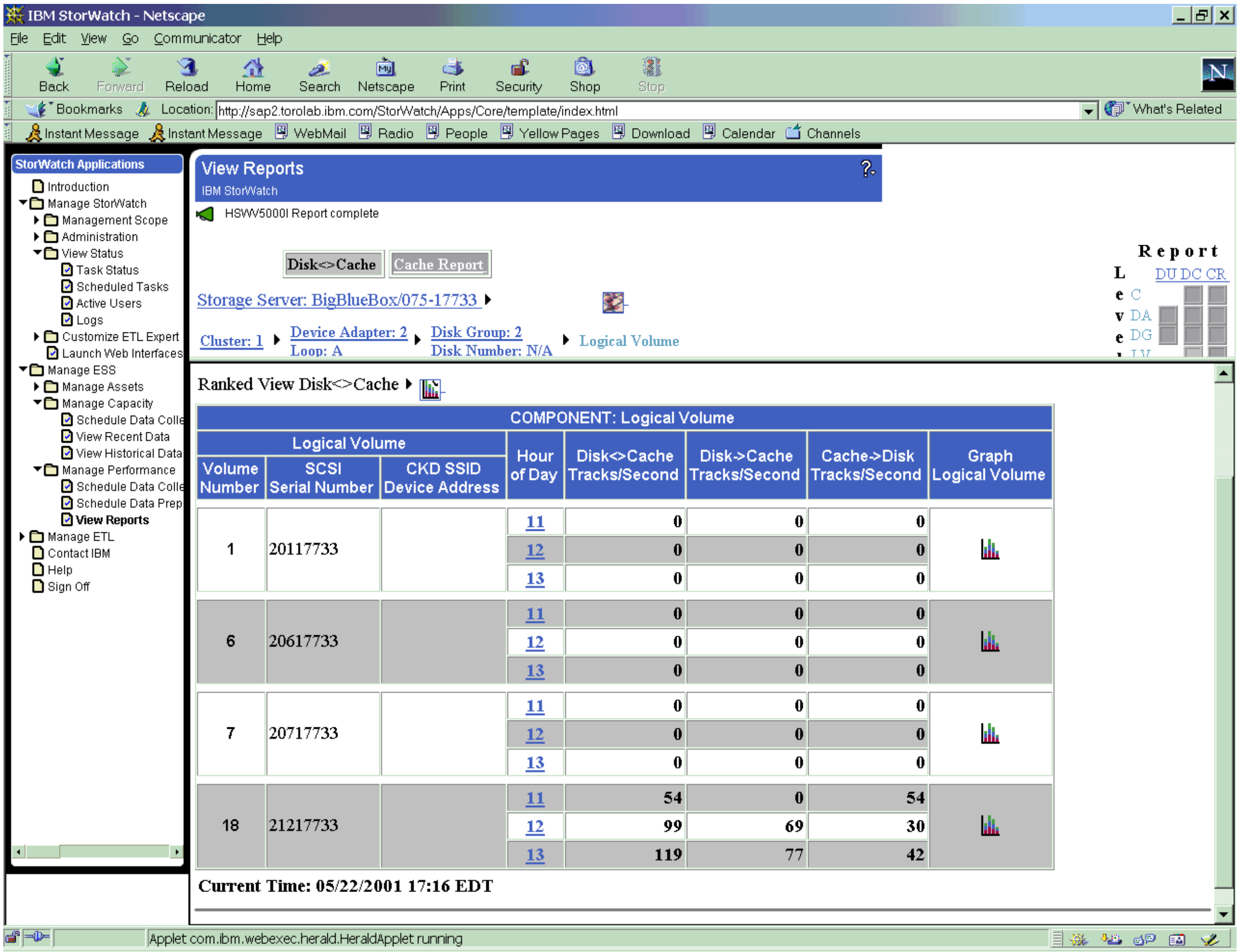


Figure 6-5 StorWatch Expert disk to/from cache report

ESS Expert — Cache reports

The Cache Summary report shows you how well cluster cache works for your application. Like the other two Summary reports if you would like to see a detail report for a specific day, you simply click on the date you want to review. The cache report summary can be seem in Figure 6-6.

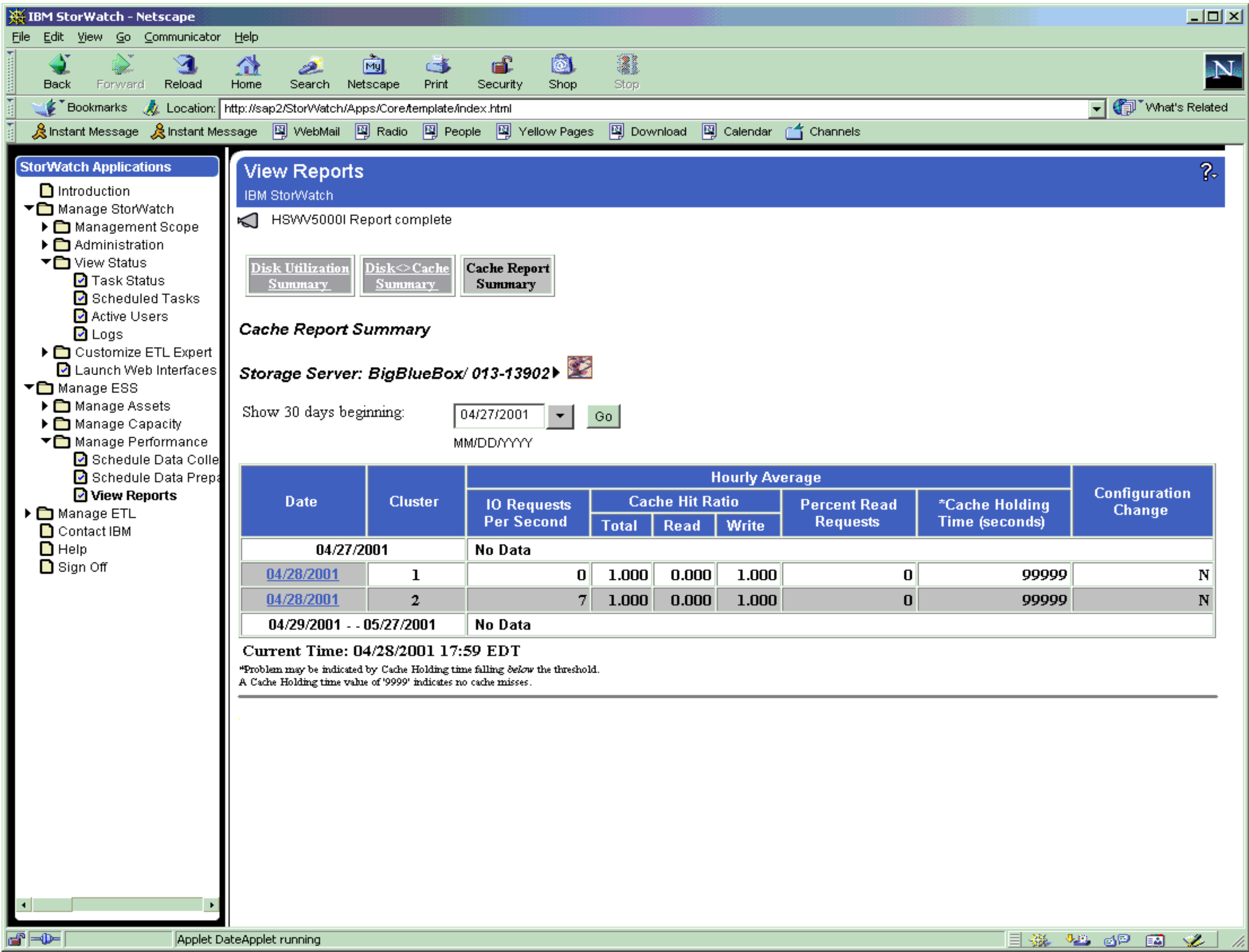


Figure 6-6 ESS Expert cache report summary

After looking at the summary report we can drill down at look at the report at a logical volume level, see Figure 6-7.

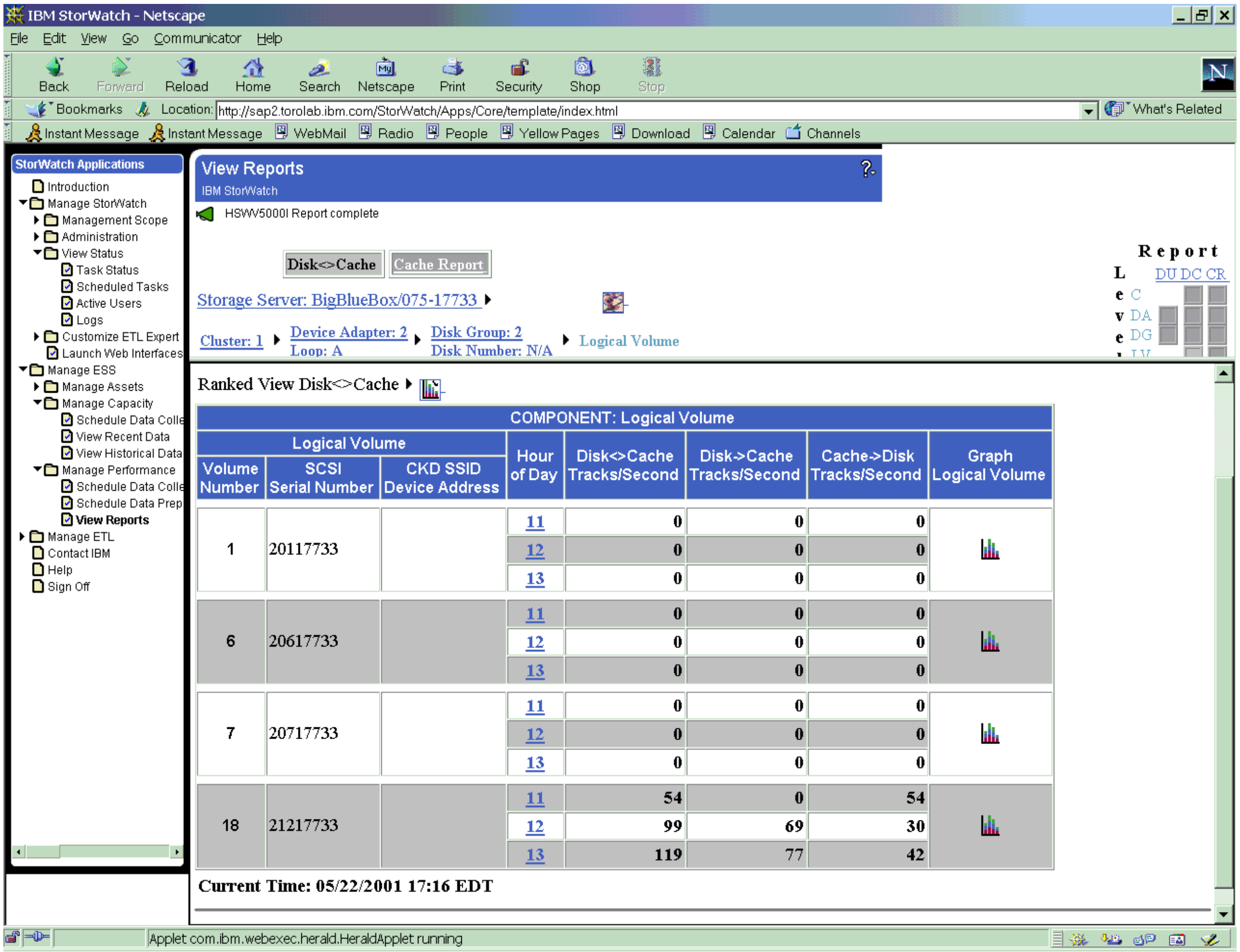


Figure 6-7 ESS Expert cache report, logical volume level

6.1.3 ESS performance tools summary

For tracking how well the ESS is performing within itself, and how well it is utilizing its cache, the ESS Expert tools are useful. However, data is not presented in terms of Mb per second or transactions per second, which is a useful piece of information to have when trying to diagnose performance issues. In addition, data is always summarized over a minimum of a five minute interval, depending on the performance issue you are trying to address, this time interval may not be granular enough. For that, we need to turn to the operating system performance tools.

6.2 AIX performance and diagnostic tools

The AIX operating system provides a variety of commands and tools to monitor the system resource activities. We will introduce some of the commands that are most useful for examining the performance of our Enterprise Database Server.

First we will discuss the capabilities of two software packages that significantly enhance AIX's capabilities. These are the Subsystem Device Driver (SDD), and the ESS utility package (ESSutils) and are used only when an Enterprise Storage Server has been installed.

6.2.1 Subsystem Device Driver (SDD) tools

If you are implementing multipathing, you will need to install the IBM subsystem device driver software (SDD). From a simplified perspective, this software resides as an additional layer between the operating system and the standard disk drivers. It is used to implement load balancing and failover across multiple paths. It also presents a single image of the disk to the host logical volume manager, where it would otherwise see multiple views of each disk - one for each path.

From a tuning standpoint, the SDD can be configured to schedule virtual path usage using the default load-balancing scheme, a round-robin scheme, or failover-only. This last option reserves a path to be used only when other paths become unavailable. Please refer to the *IBM Subsystem Device Driver Installation and User's Guide* for complete information. This document can be downloaded from the following Web site:

`http://ssddom02.storage.ibm.com/disk/ess/related.html`

The easiest way to use SDD commands on AIX platforms is through the System Management Interface Tool, SMIT, but several command-line commands are available. Of these, five are of interest here. Three map SDD virtual paths (vpaths) or AIX hdisks to ESS logical volumes, and two provide performance statistics.

List virtual path configuration

The most commonly used SDD command is **`lsvpcfg`**, which returns a listing of virtual paths, their availability, the unit serial number of the ESS logical volume itself (referred to as the ESS LUN in the SDD documentation), the AIX hdisks assigned to the logical volume, the status of each hdisk, and the name of the AIX logical volume group to which the hdisk belongs. The first three characters of the unit serial number field, the 018 of the 018FCA28 field in Figure 6-8, are the Volume id in the ESS Storwatch tabular output. The remainder of the unit serial number identifies the ESS cabinet. The "pv" stands for "physical volume," and normally should not exist in both the vpath and hdisk "Avail" field for the same vpath.

```
>lsvpcfg
vpath0 (Avail ) 018FCA28 = hdisk50 (Avail pv sapdata1vg) hdisk86 (Avail pv sapdata1vg)
vpath1 (Avail ) 019FCA28 = hdisk51 (Avail pv sapdata2vg) hdisk87 (Avail pv sapdata2vg)
vpath2 (Avail ) 01AFCA28 = hdisk52 (Avail pv sapdata3vg) hdisk88 (Avail pv sapdata3vg)
vpath3 (Avail ) 418FCA28 = hdisk53 (Avail pv sapdata1vg) hdisk89 (Avail pv sapdata1vg)
vpath4 (Avail ) 419FCA28 = hdisk54 (Avail pv sapdata2vg) hdisk90 (Avail pv sapdata2vg)
vpath5 (Avail ) 41AFCA28 = hdisk55 (Avail pv sapdata3vg) hdisk91 (Avail pv sapdata3vg)
vpath6 (Avail ) 218FCA28 = hdisk56 (Avail pv sapdata4vg) hdisk92 (Avail pv sapdata4vg)
vpath7 (Avail ) 219FCA28 = hdisk57 (Avail pv sapdata5vg) hdisk93 (Avail pv sapdata5vg)
vpath8 (Avail ) 21AFCA28 = hdisk58 (Avail pv sapdata6vg) hdisk94 (Avail pv sapdata6vg)
vpath9 (Avail ) 618FCA28 = hdisk59 (Avail pv sapdata4vg) hdisk95 (Avail pv sapdata4vg)
vpath10 (Avail ) 619FCA28 = hdisk60 (Avail pv sapdata5vg) hdisk96 (Avail pv sapdata5vg)
vpath11 (Avail ) 61AFCA28 = hdisk61 (Avail pv sapdata6vg) hdisk97 (Avail pv sapdata6vg)
vpath12 (Avail ) 01BFCA28 = hdisk62 (Avail pv logvg) hdisk98 (Avail pv logvg)
```

Figure 6-8 The lsvpcfg command

Datapath query device

Another way to look at device information is to use the **`datapath query device`** command. It displays information about each actual path assigned to a virtual path. For example, Figure 6-9 shows that virtual path vpath0 has only one physical path via Fibre-Channel adapter fscsi0/hdisk6.

```
>datapath query device 0

DEV#:   0  DEVICE NAME: vpath0  TYPE: 2105F20   SERIAL: 70016229
================================================================
Path#             Adapter/Hard Disk    State     Mode     Select     Errors
    0                 fscsi0/hdisk6    CLOSE   NORMAL          0          0
```

Figure 6-9   Datapath query device

Datapath query device statistics

Limited statistics for devices can be obtained via the **datapath query devstats** command, as illustrated in Figure 6-10. Note that these statistics have no meaningful time component. They reveal only totals since path creation (Total Read, Total Write), or instantaneous values (Active Read, Active Write).

```
>datapath query devstats 0

Device #:   0
=============
                Total Read  Total Write  Active Read  Active Write   Maximum
I/O:                     0            0            0             0         0
SECTOR:                  0            0            0             0         0

Transfer Size:      <= 512        <= 4k       <= 16K        <= 64K     > 64K
                         0            0            0             0         0
```

Figure 6-10   Datapath query devstats

Datapath query adapter

The datapath query adapter command can be used to quickly determine the health of individual data paths, as shown in Figure 6-11. However, interpretation of this output can become more confusing if you are using a switched-fabric network.

```
>datapath query adapter

Active Adapters :8

Adpt#     Adapter Name   State     Mode     Select     Errors  Paths  Active
    0           fscsi0  NORMAL   ACTIVE          0          0     13       0
    1           fscsi1  NORMAL   ACTIVE          0          0     13       0
    2           fscsi2  NORMAL   ACTIVE          0          0     13       0
    3           fscsi3  NORMAL   ACTIVE          0          0     13       0
    4           fscsi4  NORMAL   ACTIVE          0          0     13       0
    5           fscsi5  NORMAL   ACTIVE          0          0     13       0
    6           fscsi6  NORMAL   ACTIVE          0          0     13       0
    7           fscsi7  NORMAL   ACTIVE          0          0     13       0
```

Figure 6-11   Datapath query adapter

Datapath query adapter statistics

Finally, you can see total and instantaneous statistics at the adaptor level as well, using the **datapath query adaptstats** command, as illustrated in Figure 6-12.

```
>datapath query adaptstats

Adapter #:  0
=============
                Total Read  Total Write  Active Read  Active Write   Maximum
I/O:                     0            0            0             0         0
SECTOR:                  0            0            0             0         0

Adapter #:  1
=============
                Total Read  Total Write  Active Read  Active Write   Maximum
I/O:                     0            0            0             0         0
SECTOR:                  0            0            0             0         0
```

Figure 6-12 Datapath query adaptstats

The **lsvpcfg** command runs rather slowly. If you are confident that your configuration hasn't changed recently, you can get much of the same information by querying the AIX ODM database directly, as shown in Figure 6-13. This is done by using the **odmget** command.

```
>odmget -q\
   "name like vpath* and attribute = active_hdisk and value like hdisk* "\
   CuAt \
| awk '( $1 == "name"  ) { VPATH = substr($3,2,length($3)-2) }
       ( $1 == "value" ) { HDISK = substr($3,2,length($3)-11)
                           LUN   = substr($3,length(HDISK)+3,8) ;
                           printf ("%s\t%s\t%s\n", VPATH, HDISK, LUN) }' \
| sort -k1.6n,1 -k2.7n,2 -k3,3

vpath0  hdisk60        00117733
vpath0  hdisk186       00117733
vpath0  hdisk312       00117733
vpath1  hdisk61        00017733
vpath1  hdisk187       00017733
vpath1  hdisk313       00017733
vpath2  hdisk62        20017733
vpath2  hdisk188       20017733
vpath2  hdisk314       20017733
```

Figure 6-13 Querying the AIX ODM for virtual path information

6.2.2 The ESSutil package

If you have access to the IBM intranet, you can download the ESSutil utilities available as a tar file from the following Web site:

`http://enoch.tucson.ibm.com/support/storwatch/sdd_miscellaneous_links.htm`

Be sure to download the white paper *IBM Subsystem Device Driver/Data Path Optimizer on an ESS* as well. It describes the use of the utilities. This package is a must since it includes software that automates the linking of ESS logical disks to AIX hdisks. This gives you the most comprehensive view of your storage across the system. Figure 6-14 shows the output from the **lsess** command.

```
 Disk        Location       LUN SN       Type        Size    LSS    Vol    Rank   S    Connection
------      ------------   --------  ------------   ----    ---    ---    ----   -   -----------
hdisk114    B0-60-01[FC]   60617733  IBM 2105-F20   12.6     22      6    1600  04             Y
hdisk115    B0-60-01[FC]   60817733  IBM 2105-F20   12.6     22      8    1601  04             Y
hdisk116    B0-60-01[FC]   60A17733  IBM 2105-F20   12.6     22     10    1602  04             Y
hdisk117    B0-60-01[FC]   60C17733  IBM 2105-F20   12.6     22     12    1603  04             Y
hdisk118    B0-60-01[FC]   60E17733  IBM 2105-F20   12.6     22     14    1604  04             Y
hdisk119    B0-60-01[FC]   61017733  IBM 2105-F20   12.6     22     16    1605  04             Y
```

Figure 6-14 The lsess command output

Figure 6-15 shows the output from the lsdd command.

```
Hostname   VG         vpath    hdisk    Location       LUN SN   S  Connection  Size  LSS  Vol  Rank
--------   --         -----    -----    --------       ------   -  ----------  ----  ---  ---  ----
bigbluebox essdatavg1 vpath0   hdisk60  B0-60-01      00117733 Y  R1-B1-H4-ZA 122.7 16   1    1001
bigbluebox essdatavg1 vpath0   hdisk186 30-60-01      00117733 Y  R1-B2-H4-ZA 122.7 16   1    1001
bigbluebox essdatavg1 vpath0   hdisk312 90-60-01      00117733 Y  R1-B4-H4-ZA 122.7 16   1    1001

bigbluebox essdatavg1 vpath1   hdisk61  B0-60-01      00017733 Y  R1-B1-H4-ZA 105.2 16   0    1000
bigbluebox essdatavg1 vpath1   hdisk187 30-60-01      00017733 Y  R1-B2-H4-ZA 105.2 16   0    1000
bigbluebox essdatavg1 vpath1   hdisk313 90-60-01      00017733 Y  R1-B4-H4-ZA 105.2 16   0    1000

bigbluebox essda3vg1  vpath2   hdisk62  B0-60-01      20017733 Y  R1-B1-H4-ZA 50.4  18   0    1202
bigbluebox essda3vg1  vpath2   hdisk188 30-60-01      20017733 Y  R1-B2-H4-ZA 50.4  18   0    1202
bigbluebox essda3vg1  vpath2   hdisk314 90-60-01      20017733 Y  R1-B4-H4-ZA 50.4  18   0    1202
```

Figure 6-15 The lssdd command output

6.2.3 AIX diagnostic tools

The AIX operating system provides a number of tools that can be used to view system activity. We will describe here the most commonly used ones.

iostat

From an AIX operating system perspective, the best tool for investigating I/O performance is iostat. With this tool you can monitor I/O statistics at both the individual hdisk level and aggregated. The output indicates how active your CPUs are as they issue I/O commands and wait for their completion, and you can view the performance of each ESS logical disk. Figure 6-16 is output from a typical iostat listing.

```
>iostat -d hdisk288 hdisk414 hdisk162 30 5

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk288         5.1     454.3      6.7   12042630   9143565
hdisk414         5.1     449.0      6.7   11883078   9054305
hdisk162         5.0     505.3      7.4   13762027   9799044
hdisk288        50.3     3611.0    164.4      33336     75004
hdisk414        47.8     3457.7    162.2      29492     74248
hdisk162        46.1     3765.6    176.3      31676     81304
hdisk288        32.9     2352.5    119.9      15160     55420
hdisk414        27.3     2108.4    111.9      11132     52124
hdisk162        30.1     2442.4    119.6      18352     54924
hdisk288        17.5     1146.6     70.2        920     33480
hdisk414        17.2     1231.1     76.5        384     36552
hdisk162        13.9     1009.5     61.6       1084     29204
hdisk288        25.6     1882.6     74.1          8     56476
hdisk414        26.2     1937.6     75.9          4     58128
hdisk162        25.0     2007.6     82.7          8     60224
```

Figure 6-16 A sample iostat listing

The first group of output lines is a summary of all activity since the last system boot. The remaining lines are grouped by each time interval measured.

If you are using multipathing, the true statistic for a particular ESS logical disk will be the sum of the numbers for every AIX system hdisk that is mapped to the ESS logical disk. That is, the figures returned by **iostat** are for each path to the ESS logical disk, not the logical disk itself. As explained in the Mapping chapter, you will need to use the **lsvpcfg** command to determine which hdisks to summarize for a particular ESS logical disk. For instance, assuming that the three hdisks reported in Figure 6-16 are the hdisks comprising a single vpath, vpath102, we could interpret the output as shown in Figure 6-17.

```
Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk288         5.1     454.3      6.7   12042630   9143565
hdisk414         5.1     449.0      6.7   11883078   9054305
hdisk162         5.0     505.3      7.4   13762027   9799044
vpath102:       ----    1408.6     20.8   37687735  27996914

hdisk288        50.3     3611.0    164.4      33336     75004
hdisk414        47.8     3457.7    162.2      29492     74248
hdisk162        46.1     3765.6    176.3      31676     81304
vpath102:       ----    10834.3    502.9      94504    230556

.
.
.
```

Figure 6-17 Interpreted iostat listing for two disks associated with a single vpath

Note that with a RAID5 implementation, a value of 100 in the % tm_act column of the iostat output does not mean that the RAID array physical disks are 100% busy. This column is the **iostat**'s interpretation of the percent of time the individual path to the ESS logical volume is busy.

To determine the **transfer block** size, divide Kbps (kilobytes per second) by tps (transfers per second). To estimate the **read/write ratio** of the ESS logical disk, divide Kb_read by Kb_wrtn.

vmstat

You can use the **vmstat** command to report statistics about kernel threads in the run and wait queues, memory, paging, disks, interrupts, system calls, context switches, and CPU activity. If the **vmstat** command is used without any options, or only with the interval and/or count option, then the first line of numbers is an average since the system reboot.

You can use `vmstat` output lines to identify deviations over time which may be indicative of virtual memory problems. In particular, look for marked increases of pages paged in or out. You may also see increases in device interrupts, system calls, and context switches. Low CPU idle times may indicate that your application is compute-intensive with insufficient cycles available for I/O. Finally, high wait times may indicate unusually high demand on your virtual memory devices. Figure 6-18 is a sample `vmstat` output.

```
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 87360 3161051   0   0   0  58  105   0 132   369 169 12  4 75  9
 0  4 85150 3163595   0   0   0   0    0   0 2476 8534 207  1  1 98  0
 1  4 85628 3162817   0   0   0   0    0   0 2510 17963 619  3  3 94  0
 0  4 85002 3163764   0   0   0   0    0   0 2417 13762  90  0  2 98  0
 0  4 85002 3163764   0   0   0   0    0   0 2412   439  39  0  0 99  0
```

Figure 6-18 Sample vmstat output

filemon

The filemon utility is a great tool for ensuring your workload is spread evenly and identifying hot spots. It is a trace tool, which means it must be started and stopped. Trace tools also tend to consume system resources fairly heavily, so use filemon carefully. After filemon has been stopped, it will analyze its results and generate a report. You should use the -o option of the command to direct the report to a file you can examine.

Figure 6-19 shows storage related selections from a filemon output after running the tool for approximately 30 seconds. The entire output can be quite lengthy when you are looking at ESS storage. It is nicely detailed, with excellent summarizations. In addition to showing storage activity. As with iostat, filemon output must be interpreted carefully. The activity of a single logical volume does not translate to total ESS array activity; and other activity on the array will influence the numbers you see. You'll learn to interpret the results, but begin by looking for an overall pattern. Our example illustrates a very even workload, which is what we want to see.

```
Most Active Logical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume              description
------------------------------------------------------------------------
  0.96  57920      0  847.7  /dev/lv.rb.D1.DA8c1  raw
  0.96  57920      0  847.7  /dev/lv.rb.D1.DA5b1  raw
  0.96  57856      0  846.8  /dev/lv.rb.D1.DA6d1  raw
  0.96  57920      0  847.7  /dev/lv.rb.D1.DA3b1  raw
  0.96  57920      0  847.7  /dev/lv.rb.D1.DA4e1  raw
  0.96  57856      0  846.8  /dev/lv.rb.D1.DA6a1  raw
  0.96  57856      0  846.8  /dev/lv.rb.D1.DA8a1  raw
  0.96  57856      0  846.8  /dev/lv.rb.D1.DA5d1  raw
  0.96  57920      0  847.7  /dev/lv.rb.D1.DA5a1  raw
Most Active Physical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume              description
------------------------------------------------------------------------
  0.59 127936      8 1872.6  /dev/hdisk179       IBM FC 2105F20
  0.59 122304     16 1790.3  /dev/hdisk176       IBM FC 2105F20
  0.59 130688     40 1913.4  /dev/hdisk161       IBM FC 2105F20
  0.59 129024     32 1888.9  /dev/hdisk165       IBM FC 2105F20
  0.58 129440     40 1895.1  /dev/hdisk160       IBM FC 2105F20
  0.58 128352      8 1878.7  /dev/hdisk174       IBM FC 2105F20
  0.58 125952     32 1843.9  /dev/hdisk182       IBM FC 2105F20
  0.57 105056     24 1538.0  /dev/hdisk424       IBM FC 2105F20
  0.57 127872      8 1871.7  /dev/hdisk157       IBM FC 2105F20
------------------------------------------------------------------------
Detailed Logical Volume Stats   (512 byte blocks)
------------------------------------------------------------------------
VOLUME: /dev/lv.rb.D1.DA8c1  description: raw
reads:      1810(0 errs)
  read sizes (blks): avg    32.0 min      32 max      32 sdev     0.0
  read times (msec):avg  43.261 min  32.991 max 524.179 sdev  39.365
  read sequences: 7
  read seq. lengths:avg  8274.3 min      64 max   28864 sdev 11853.9
seeks:      7   (0.4%)
  seek dist (blks):init 260288,
           avg    85.3 min      64 max     128 sdev    30.2
time to next req(msec): avg  18.864 min   0.001 max 536.519 sdev  28.021
throughput:847.7 KB/sec
utilization:0.96
------------------------------------------------------------------------
Detailed Physical Volume Stats   (512 byte blocks)
------------------------------------------------------------------------
VOLUME: /dev/hdisk179  description: IBM FC 2105F20
reads:      2000(0 errs)
  read sizes (blks): avg    64.0 min      32 max      64 sdev     1.0
  read times (msec):avg  17.047 min   0.053 max 516.848 sdev  28.503
  read sequences: 1902
  read seq. lengths:avg    67.3 min      32 max     192 sdev    14.3
writes:     1   (0 errs)
  write sizes (blks): avg     8.0 min       8 max       8 sdev     0.0
  write times (msec):avg   4.685 min   4.685 max   4.685 sdev   0.000
  write sequences: 1
  write seq. lengths:avg     8.0 min       8 max       8 sdev     0.0
seeks:      1903(95.1%)
  seek dist (blks):init 8115008,
           avg 18311324.2 min      64 max 41941632 sdev 10162236.0
  seek dist (%tot blks):init 8.82141,
           avg 19.90530 min 0.00007 max 45.59260 sdev 11.04685
time to next req(msec): avg  17.070 min   0.066 max 505.553 sdev  23.220
throughput:1872.6 KB/sec
utilization:0.5
```

Figure 6-19   Storage related selections from a filemon output

Figure 6-20 shows the the utility's capability to show CPU utilization and virtual memory.

```
Sat Jun  9 18:17:24 2001
System: AIX bigbluebox Node: 4 Machine: 000B03BD4C00

Cpu utilization:  29.5%

Most Active Segments
------------------------------------------------------------------------
  #MBs  #rpgs  #wpgs  segid  segtype              volume:inode
------------------------------------------------------------------------
   0.3      1     77   2ac0ab  page table
   0.1      0     24   3080c2  page table
   0.1      0     17   2e00b8  log
.
------------------------------------------------------------------------
Detailed VM Segment Stats   (4096 byte pages)
------------------------------------------------------------------------
SEGMENT: 2ac0ab  segtype: page table
segment flags:pers log pgtbl
reads:      1   (0 errs)
  read times (msec):avg  13.766 min  13.766 max  13.766 sdev   0.000
  read sequences: 1
  read seq. lengths:avg     1.0 min       1 max       1 sdev     0.0
writes:    77  (0 errs)
  write times (msec):avg 269.993 min 155.033 max 450.779 sdev  79.250
  write sequences: 77
  write seq. lengths:avg     1.0 min       1 max       1 sdev     0.0
```

Figure 6-20 CPU and virtual memory related selections from a filemon output

The topas tool

The interactive AIX tool, **topas**, is convenient if you want to get a quick overall view of the system's current activity. A fast snapshot of memory usage or user activity can be a helpful starting point for further investigation. However, **topas** is of very limited use as a diagnostic tool when you are dealing with a large number of logical volumes on an ESS. Figure 6-21 is a sample **topas** screen.

```
Topas Monitor for host:    bigbluebox          EVENTS/QUEUES    FILE/TTY
Thu May 31 10:44:28 2001   Interval:  2        Cswitch     528  Readch    28532
                                               Syscall    1269  Writech     140
Kernel    0.1   |                            | Reads       104  Rawin         0
User      0.8   |                            | Writes        3  Ttyout       94
Wait     40.7   |###########                 | Forks         2  Igets         0
Idle     58.2   |################            | Execs         2  Namei        42
                                               Runqueue    0.0  Dirblk        0
Interf   KBPS   I-Pack  O-Pack   KB-In  KB-Out Waitqueue  12.0
en2        0.6     2.0     2.5     0.2     0.4
lo0        0.0     0.0     0.0     0.0     0.0 PAGING           MEMORY
                                               Faults      321  Real,MB   49151
Disk    Busy%     KBPS     TPS KB-Read KB-Writ Steals        0  % Comp     18.2
hdisk1    0.0      0.0     0.0     0.0     0.0 PgspIn        0  % Noncomp  11.9
hdisk2    0.0      0.0     0.0     0.0     0.0 PgspOut       0  % Client    0.8
hdisk4    0.0      0.0     0.0     0.0     0.0 PageIn        0
hdisk0    0.0      0.0     0.0     0.0     0.0 PageOut       0  PAGING SPACE
hdisk3    0.0      0.0     0.0     0.0     0.0 Sios          0  Size,MB   33824
                                                                % Used      0.5
topas    (52976) 19.0% PgSp: 1.8mb perfpol2                     % Free     99.4
dtgreet  (8552)   1.0% PgSp: 1.3mb root
db2sysc  (46044)  0.5% PgSp: 0.4mb reg64
db2sysc  (37534)  0.5% PgSp: 0.4mb reg64          Press "h" for help screen.
init     (1)      0.0% PgSp: 0.6mb root           Press "q" to quit program.
```

Figure 6-21   A sample topas screen

6.3 DB2 UDB performance and diagnostics tools

DB2 UDB provides the ability to monitor database activity using tools, such as the Snapshot Monitor, the Event Monitor, and the CLI/ODBC/JDBC trace tool.

For complete instructions on how to use these tools, refer to *DB2 UDB Administration Guide: Performance V7*, SC09-2945 and *DB2 UDB System Monitor Guide and Reference*, SC09-2956, which are available online at:
`http://www-4.ibm.com/software/data/db2/library`

Another valuable resource is the *DB2 UDB V7.1 Performance Tuning Guide*, SG24-6012. which can be found online at: `http://www.redbooks.ibm.com`.

6.3.1 The Snapshot Monitor

The Snapshot Monitor is used to capture performance information at points of time. Information is kept as a point-in-time value, such as a counter, a high water mark, or the last timestamp for a particular event/object.

The snapshot monitor keeps track of information at various levels, including database manager, databases, applications, buffer pools, table spaces, tables, locks, and statements.

Some basic information is always tracked by DB2. Additional information can be tracked by turning on one or multiple Snapshot Monitor switches. The switches available include sort, lock, table, bufferpool, uow (unit of work), and statement.

Note that the amount of memory available to the snapshot monitor is controlled by the MON_HEAP_SZ database manager configuration parameter. If you will be turning switches on, you should check how much memory has been set aside for its use.

Activating a monitor switch can be done by issuing the UPDATE MONITOR SWITCHES command from the command line interface. This enables the monitor switch for that session only. You may also enable the monitor switch permanently for the entire instance. To do this, use the “update dbm cfg using “ command.

Once a monitor switch is turned on, the “get snapshot monitor” command is used to report the collected data.

Example 6-1 Snapshot monitor output - partial listing

```
Database Snapshot

Database name                              = TEST2
Database path                              = /home/db2inst1/db2inst1/NODE0000/SQL00002/
Input database alias                       = TEST2
Database status                            = Active
Catalog node number                        = 0
Catalog network node name                  =
Operating system running at database server= AIX
Location of the database                   = Local
First database connect timestamp           = 07-27-2000 05:58:33.471961
Last reset timestamp                       =
Last backup timestamp                      = 08-14-2000 05:22:32.217719
Snapshot timestamp                         = 08-14-2000 09:00:02.540889

Total sort heap allocated                  = 1683
Total sorts                                = 27554148
Total sort time (ms)                       = 46924304
Sort overflows                             = 37880
Active sorts                               = 3

High water mark for database heap          = 41301532

Buffer pool data logical reads             = 3525982687
Buffer pool data physical reads            = 46068863
Asynchronous pool data page reads          = 43595577
Buffer pool data writes                    = 40216429
Asynchronous pool data page writes         = 39275356
Buffer pool index logical reads            = 1911184535
Buffer pool index physical reads           = 3739755
Asynchronous pool index page reads         = 2447983
Buffer pool index writes                   = 531164
Asynchronous pool index page writes        = 324272
Total buffer pool read time (ms)           = 19958794
Total buffer pool write time (ms)          = 7597768
Total elapsed asynchronous read time       = 13099308
Total elapsed asynchronous write time      = 7121334
Asynchronous read requests                 = 6715104
LSN Gap cleaner triggers                   = 14881
Dirty page steal cleaner triggers          = 245497
Dirty page threshold cleaner triggers      = 960944
Time waited for prefetch (ms)              = 5854
Direct reads                               = 542431372
Direct writes                              = 23088829
Direct read requests                       = 82397057
Direct write requests                      = 378384
Direct reads elapsed time (ms)             = 7884745
```

```
Direct write elapsed time (ms)            = 573733
Database files closed                     = 111177
Data pages copied to extended storage     = 0
Index pages copied to extended storage    = 0
Data pages copied from extended storage   = 0
Index pages copied from extended storage  = 0

Commit statements attempted               = 7586638
Rollback statements attempted             = 360826
Dynamic statements attempted              = 160387056
Static statements attempted               = 7949808
Failed statement operations               = 47008
Select SQL statements executed            = 69737342
Update/Insert/Delete statements executed  = 6616697
DDL statements executed                   = 40

Internal automatic rebinds                = 0
Internal rows deleted                     = 28665
Internal rows inserted                    = 865594
Internal rows updated                     = 145131
Internal commits                          = 698993
Internal rollbacks                        = 5946
Internal rollbacks due to deadlock        = 0

Rows deleted                              = 2054516
Rows inserted                             = 1713665
Rows updated                              = 8803116
Rows selected                             = 223993815

Binds/precompiles attempted               = 3059

Log space available to the database (Bytes)= 20395920
Log space used by the database (Bytes)    = 0
Maximum secondary log space used (Bytes)  = 0
Maximum total log space used (Bytes)      = 123690123
Secondary logs allocated currently        = 0
Log pages read                            = 14
Log pages written                         = 4164032
Appl id holding the oldest transaction    = 0

Package cache lookups                     = 76313826
Package cache inserts                     = 29236843
Package cache overflows                   = 0
Package cache high water mark (Bytes)     = 20480000
Application section lookups               = 160389400
Application section inserts               = 73166096

Catalog cache lookups                     = 58935612
Catalog cache inserts                     = 2528
Catalog cache overflows                   = 0
Catalog cache heap full                   = 0

Number of hash joins                      = 0
Number of hash loops                      = 0
Number of hash join overflows             = 0
Number of small hash join overflows       = 0
```

As you can see in Example 6-1, the snapshot monitor can report on many facets of the database activity. In addition to the partial database level information shown, the snapshot monitor can report on bufferpool usage, dynamic SQL statements and applications.

6.3.2 The Event Monitor

The Event Monitor is used to provide a summary of activity at the completion of events, such as statement execution, transaction completion, or when an application disconnects. The event monitor records the database activity every time an event or transition occurs.

Many of the events to be monitored are captured and logged to disk when the last application disconnects from the database. These include the following: database, tables, tablespaces, and bufferpools. In addition to these, the following events can also be tracked: deadlocks, connections, statements, and transactions.

Monitors are created using the “create event monitor” statement. DB2 also provides a GUI tool which can be used for this purpose (shown in Example 6-22). Filters can be specified for an event monitor to limit the event monitor data based on values, such as application ID or application name.

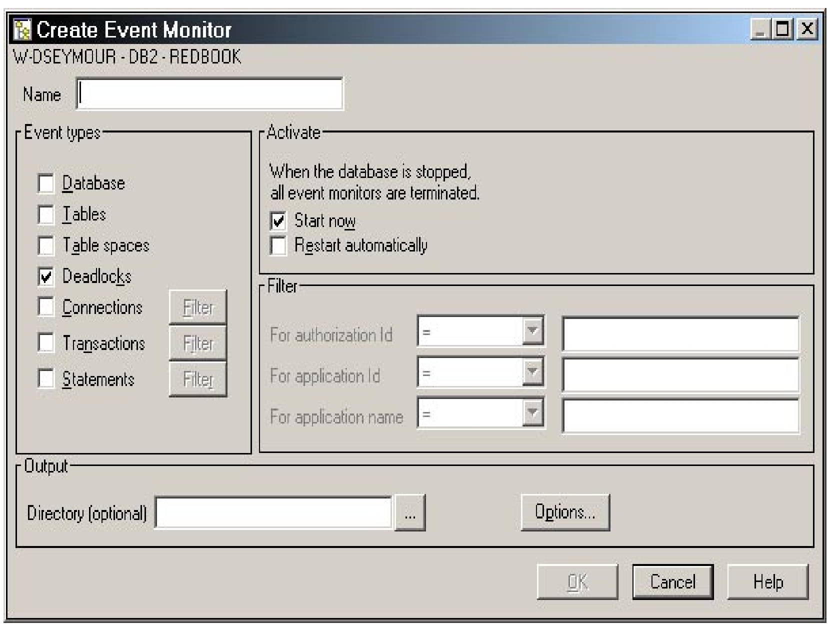


Figure 6-22 DB2 UDB Event Monitor setup

The “db2 flush event monitor” command can be used to force the monitor to generate a record even when applications are still connected. These records will be partial records and noted as such in the output.

Setting up an event monitor can be useful when trying to understand things that are difficult to capture during a snapshot, such as deadlock situations.

The db2evmon utility can be used to format the output of the event monitors. *DB2 UDB System Monitor Guide and Reference*, SC09-2956, contains examples of how to setup the Event monitor and to work with the results.

A simple example, Example 6-23, shows an Event monitor which was set up to capture data regarding connections for a particular userid, DSEYMOUR.

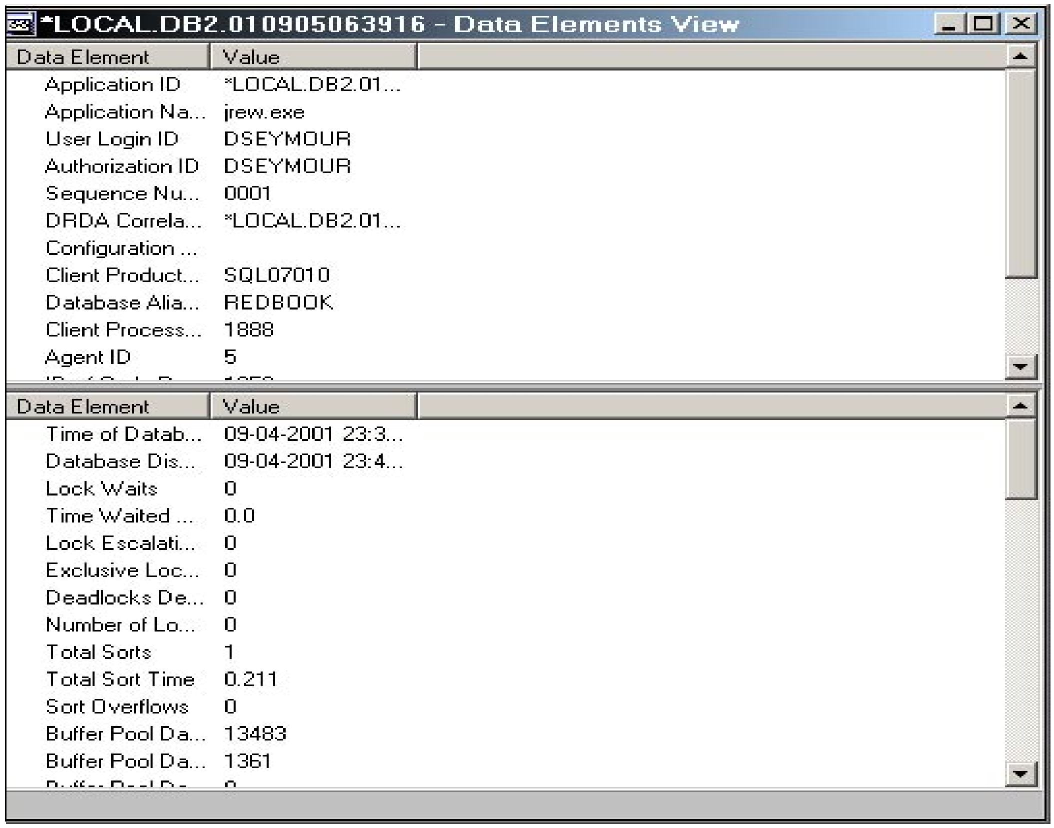


Figure 6-23 Event Monitor output

6.3.3 The Performance Monitor

DB2 also provides a graphical view of DB2 performance, called the Performance Monitor. Default Performance Monitor definitions come with DB2 UDB and can be used as is, or modified to track the Snapshot and/or Event data that you want. The performance Monitor provides you with a graphical view of the data which you can customize. It also allows you to set thresholds for parameters of your choice.

Following (Example 6-24) is an example of the Performance Monitor (this was run on Windows 2000) showing the read throughput for a query of an iostat table.

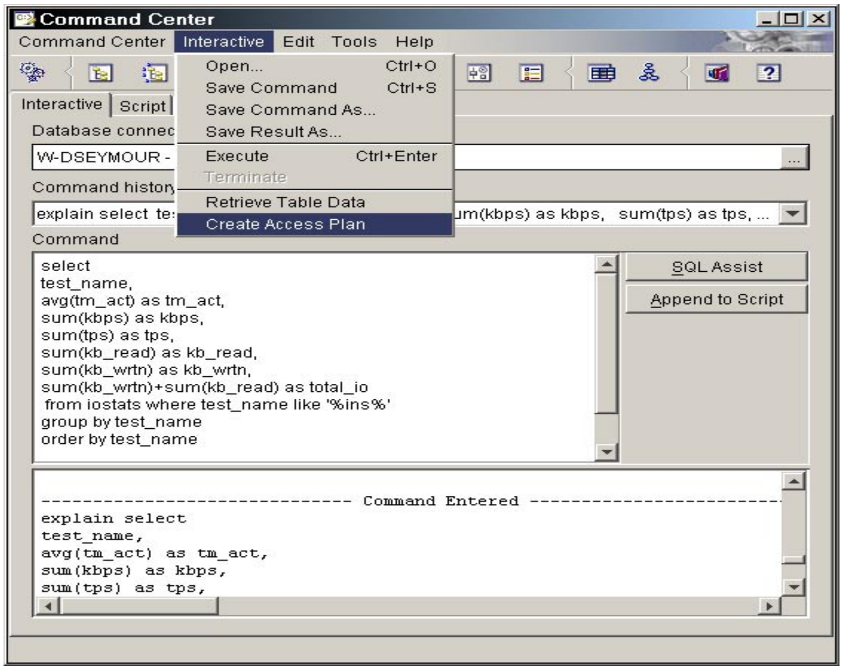


Figure 6-24 DB2 UDB Performance Monitor

The performance monitor is launched from the DB2 Control Center. Right-clicking at the object level that you want to monitor will bring up a list of activities to choose from, selecting ‘Performance Monitoring...’ allows entry into these facilities.

Additional information on the Performance Monitor may be found in the online help for DB2 UDB.

6.3.4 The Explain Facility

The **Explain Facility** provides information about how DB2 will access the data in order to resolve the SQL statements. Each SQL statement is analyzed by DB2’s *optimizer*. The optimizer determines which *access plan* will be used to retrieve data from the tables. The access plan is made up of *access paths.* Access paths describe which method will be used to retrieve data from a particular table, such as whether indexes will be used or not.

Visual explain and db2exfmt use the explain tables to store access plan information so that users can see the decisions that the optimizer has made. Once explain information has been captured for an SQL statement by using the EXPLAIN statement, it can be queried or displayed using Visual Explain or db2exfmt.

The Explain facility (either Visual Explain or db2exfmt) will tell you if any indexes are being used, and what method of join the optimizer has chosen. The optimizer depends on the use of current statistics. The utility that must be used to set these is RUNSTATS. For static SQL statements, a rebind statement can also be issued in order to take advantage of updated statistics.

Following is an example of an SQL statement which queries a summarized iostat table. In this case, the DB2 UDB Command Center was used to ‘Create the Access Plan’.

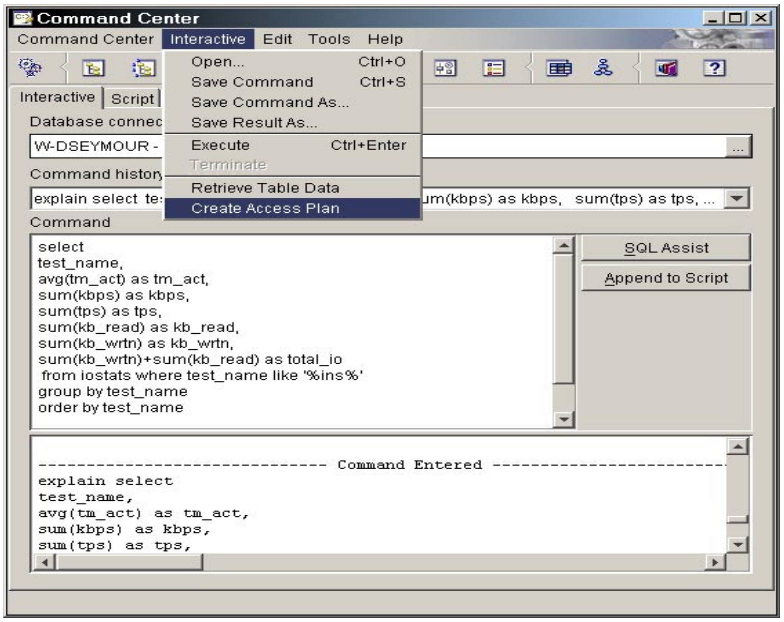


Figure 6-25 Create the access plan for an SQL statement

The result of the Access plan creation is the Visual Explain shown in Example 6-26.

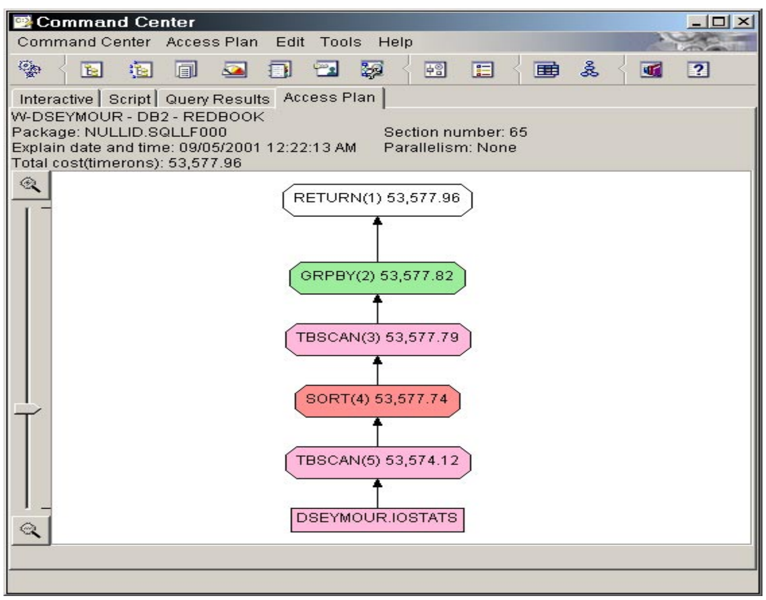


Figure 6-26 Visual Explain output

The various steps of the access plan can be drilled down on by double-clicking on the various steps of the plan.

As you can see, in this case the optimizer has chosen a tablescan for the DSEYMOUR.IOSTATS table. It is also sorting the table and grouping the results.

Changing the SQL statement to have a more specific where clause (where test_name = ‘Run98ins01’) results in an available index being used. You can also see that the cost estimate is lower. The changed access plan is shown below in Example 6-27.

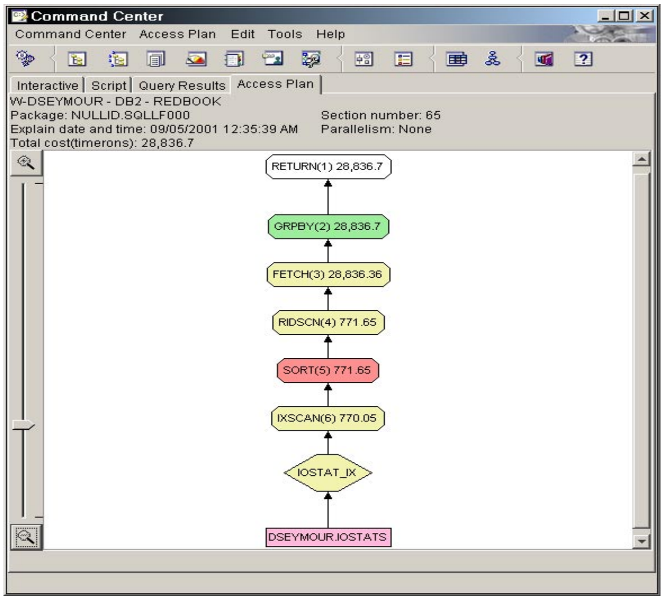


Figure 6-27 Access plan for indexed access

6.3.5 The db2batch tool

The db2batch tool can be used for benchmark testing. This tool will read SQL statements from either a flat file or standard input, dynamically describe and prepare the statements, and return an answer set. It provides the added flexibility of allowing you to control the size of the answer set, as well as the number of rows that should be sent from this answer set to an output device. You can also specify the level of performance-related information supplied, including the elapsed time, CPU and buffer pool usage, locking, and other statistics collected from the database monitor. If you are timing a set of SQL statements, db2batch will also summarize the performance results and provide both arithmetic and geometric means.

For more information on benchmark testing, please see *DB2 UDB Administration Guide: Performance V7*, SC09-2945. There are also redbooks devoted to this topic which can be found at the IBM Redbook Web site: `http://www.redbooks.ibm.com`.

6.3.6 The CLI/ODBC/JDBC Trace Facility

The trace facility traces all of the function calls of the driver (CLI, ODBC or JDBC) for use in problem determination and tuning (see *Call Level Interface Guide and Reference V7,* SC09-2950).

Obtaining a CLI trace is done by first issuing the “db2 update cli cfg” command, setting the trace options, then running the application, and finally stopping the trace and formatting the results.

The results of the trace file can be examined to show such things as:

- Time spent in the application versus time spent in DB2
- How long a particulates CLI call is taking
- Find the longest intervals of execution
- Find the number of CLI calls of various types
- Find out how much data is transferred
- Find the timing relationships between multiple threads in an application

6.4 An example of how to use the tools at hand

The example is a new system and application. The ESS has been set up, the AIX system is installed, and the database has been defined and is using DB2 UDB EE (so you know that you have a single data partition). The next step is to load data into the tables.

The source files are in a directory called “/db2data/sourcefiles”. The data contained in these files will be loaded into the “newapp” database. The database is made up of two fairly large tables, and eight smaller tables.

The required commands to load the data are saved in a file, “loaddata.sql”. This file is stored in your home directory.

To test this process, first load the smaller tables. Edit the loaddata.sql and comment out the two larger table’s load commands. Set up the following job which will execute iostats while the load job is running. It will also execute a second script, nextjob.ksh. This allows the ability to run another job directly upon completion of the current job. If you do not want to have another job kick off directly after this one, you simply do not create a job “nextjob.ksh”. This is shown in Example 6-2.

Example 6-2 loaddata.ksh

```
#!/usr/bin/ksh

# start iostats
echo “Start load job “ ‘date‘ > iostat.out
echo “30 second intervals “ >> iostat.out
iostat 30 >> iostat.out

#load data
echo “Start load “ ‘date‘ > loaddata.out
db2 -tvf loaddata.sql >> loaddata.out

#stop iostat process
zap -f iostat

nextjob.ksh
```

The iostats that were produced as part of the load are shown in Example 6-3.

Example 6-3 iostat during Load

```
tty:      tin        tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0         0.0              44.8      8.1       29.6      17.6

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
...
hdisk90          4.0     972.7       7.8         0     58368
hdisk91          4.2     972.7       7.8         0     58368
hdisk92          4.2     985.5       7.9         0     59136
hdisk93          4.1     998.6       8.0         0     59920
hdisk94          5.3     998.1       8.0         0     59888
hdisk95          4.1     985.5       7.9         0     59136
hdisk224         1.2     266.1       2.1         0     15968
hdisk231         1.0     238.6       1.9         0     14320
hdisk238         0.9     212.2       1.7         0     12736
hdisk203         1.3     274.4       2.2         0     16464
hdisk210         1.0     234.7       1.9         0     14084
hdisk217         1.2     261.3       2.1         0     15680
hdisk183         1.3     280.2       2.2         0     16816
```

```
hdisk189          1.2     290.1       2.3          0     17408
hdisk196          1.1     270.9       2.1          0     16256
hdisk96           0.0       0.0       0.0          0         0
hdisk97           0.0       0.0       0.0          0         0
hdisk245          0.0       0.0       0.0          0         0
hdisk246          0.0       0.0       0.0          0         0
hdisk247          0.0       0.0       0.0          0         0
hdisk248          0.0       0.0       0.0          0         0
hdisk228          1.0     246.6       1.9          0     14800
hdisk235          1.1     276.5       2.2          0     16592
hdisk242          1.0     234.6       1.9          0     14080
hdisk207          1.1     260.5       2.1          0     15632
```

Load will generally write evenly across all of the containers in a tablespace. Here, the variance of the measurements (from 998 kbps down to 280 kbps for the same type of write activity) are a concern.

The hdisk activity can be grouped by relative activity. The three groups include: no activity, low activity (Kbps in the 200's), and higher activity (Kbps in the 900's). One hdisk from each activity grouping should be investigated: hdisk90, hdisk96, and hdisk226 will be looked into more closely.

The first requirement is to determine which ESS logical disk these hdisks represent. Issuing an lsvpcfg command on AIX will give a listing of the vpaths and their associated hdisks.

Example 6-4 shows the commands issued to obtain vpath information for two of the hdisks.

Example 6-4 Finding the ESS logical disk for an hdisk

```
lsvpcfg > lsvpcfg.Apr29.out

grep hdisk90 lsvpcfg.Apr29.out
vpath48 (Avail ) 41213902 = hdisk90 (Avail pv ssa5vg3)

grep hdisk224 lsvpcfg.Apr29.out
vpath14 (Avail pv ssa8vg3) 71413902 = hdisk160 (Avail ) hdisk224 (Avail ) hdisk225 (Avail )
hdisk226 (Avail ) hdisk227 (Avail ) hdisk228 (Avail ) hdisk229 (Avail ) hdisk230 (Avail )
```

This information shows that hdisk90 is a single path defined to the ESS through vpath48, while hdisk224 is one of eight paths to the ESS through vpath14. Now you know that in order to find out how the two vpaths compare, you must first sum the activity on these eight paths: hdisk160, hdisk224, hdisk225, hdisk226, hdisk227, hdisk228, hdisk229 and hdisk230.

Reviewing these hdisks for the same time interval in the iostat.out file reveals the information shown in Example 6-5.

Example 6-5 iostats during Load for selected hdisks

```
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0          0.0              44.8      8.1       29.6      17.6

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn

hdisk226          0.0       4.3       0.0          0       256
hdisk90           4.0     972.7       7.8          0     58368
hdisk224          1.2     266.1       2.1          0     15968
hdisk228          1.0     246.6       1.9          0     14800
hdisk225          0.0      13.6       0.1          0       816
hdisk227          0.9     202.4       1.6          0     12144
```

```
hdisk229         0.0       8.5       0.1          0       512
hdisk160         0.8     179.5       1.4          0     10768
hdisk230         0.0       0.3       0.0          0        16
```

For Example 6-5, we simply removed the lines which contained hdisks we were not interested in. (As you can see, the hdisks are not in sequence. They are also out of sequence in the original output.)

Summarizing the information shown in the iostats output, we can see how the vpaths compare in Example 6-6.

Example 6-6 Summary of iostat by vpath

```
Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn

vpath48          4.0     972.7       7.8          0     58368
vpath14          3.9     921.3       7.2          0     55280
```

You may be concerned that the vpaths, especially vpath14 which is servicing 8 ESS logical disks, are causing a bottleneck. However vpath48, which services a single logical disk actually moves more data than vpath14. From this you can deduce that vpath14 is not causing a bottleneck.

You decide to review the iostat data for read activity. Assuming that since this read activity is of a sequential nature, you should be getting optimal performance from the reads. During the same time interval as the write activity just documented, you observe:

Example 6-7 Read iostat activity during LOAD

```
Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk98         11.9     290.8      65.5      17452         0
hdisk99         18.0     471.8     100.0      28312         0
hdisk101        12.1     329.8      69.8      19788         0
hdisk102        20.9     417.7      91.1      25064         0
hdisk103        15.4     225.2      48.4      13512         0
hdisk104        30.3     772.4     163.4      46348         0
hdisk105         5.2     101.1      21.0       6064         0
hdisk106         0.2       3.5       0.8        212         0
hdisk107         2.3      38.6       8.2       2316         0
hdisk108        10.8     104.7      26.0       6280         0
hdisk109        17.8     402.5      88.3      24152         0
```

You also see three other 'groups' of read activity similar to the group shown in Example 6-7 during the same time interval from your iostat output. From this information you are assuming that there are four paths defined to these ESS logical disks.

There are two tasks which must be accomplished here. First, as with the write activity just reviewed, you should match these hdisks to their underlying ESS logical disks. Second, you need to map these hdisks to the underlying AIX file systems to ensure that the read activity is due to reading the source files, and not to some other activity on the system.

A review of the lsvpcfg output reveals the following (which confirm your hypothesis regarding four paths enabled for these ESS logical disks):

Example 6-8 Virtual paths for Read activity

```
vpath56 (Avail pv datavg) 00013902 = hdisk98 (Avail ) hdisk249 (Avail ) hdisk271 (Avail )
hdisk285 (Avail )
vpath57 (Avail pv datavg) 00213902 = hdisk99 (Avail ) hdisk251 (Avail ) hdisk273 (Avail )
hdisk287 (Avail )
```

```
vpath58 (Avail pv datavg) 00113902 = hdisk100 (Avail ) hdisk250 (Avail ) hdisk272 (Avail )
hdisk286 (Avail )
vpath59 (Avail pv datavg) 00413902 = hdisk101 (Avail ) hdisk253 (Avail ) hdisk275 (Avail )
hdisk289 (Avail )
vpath60 (Avail pv datavg) 00613902 = hdisk102 (Avail ) hdisk255 (Avail ) hdisk277 (Avail )
hdisk291 (Avail )
vpath61 (Avail pv datavg) 00713902 = hdisk103 (Avail ) hdisk256 (Avail ) hdisk278 (Avail )
hdisk292 (Avail )
vpath62 (Avail pv datavg) 00813902 = hdisk104 (Avail ) hdisk257 (Avail ) hdisk279 (Avail )
hdisk293 (Avail )
vpath63 (Avail pv datavg) 00913902 = hdisk105 (Avail ) hdisk258 (Avail ) hdisk280 (Avail )
hdisk294 (Avail )
vpath64 (Avail pv datavg) 00313902 = hdisk106 (Avail ) hdisk252 (Avail ) hdisk274 (Avail )
hdisk288 (Avail )
vpath65 (Avail pv datavg) 00513902 = hdisk107 (Avail ) hdisk254 (Avail ) hdisk276 (Avail )
hdisk290 (Avail )
vpath66 (Avail pv datavg) 00A13902 = hdisk108 (Avail ) hdisk259 (Avail ) hdisk281 (Avail )
hdisk295 (Avail )
vpath67 (Avail pv datavg) 00B13902 = hdisk109 (Avail ) hdisk260 (Avail ) hdisk282 (Avail )
hdisk296 (Avail )
```

All of these ESS Logical disks are associated with the 'datavg' volume group. By issuing an lsvp -p command for one of these vpaths, the following information is obtained:

Example 6-9 lspv command for iostat read activity

```
vpath56:
PP RANGE  STATE   REGION        LV NAME            TYPE       MOUNT POINT
  1-11    used    outer edge    lv.kwaitemp        jfs        /db2data
 12-12    used    outer edge    loglv10            jfslog     N/A
 13-36    free    outer edge
 37-72    free    outer middle
 73-107   free    center
108-116   free    inner middle
117-142   used    inner middle  lv.kwaitemp        jfs        /db2data
143-175   used    inner edge    lv.kwaitemp        jfs        /db2data
176-178   free    inner edge
```

From this you can see that your input files are indeed the source of this read activity.

One of your concerns as a result of your observations is to ensure that all of the ESS logical disks that you are working with have been defined to use the same number of virtual paths. Greater throughput should be realized when multiple paths are defined, and you want an even distribution of resources.

6.4.1 Summary of tools used

For the most part, the analysis of this situation was based on configuration information which maps the ESS resource to DB2 (as described in the Mapping chapter of this book) in conjunction with I/O statistics as reported by AIX.

As described in Chapter 5, "Mapping ESS resources to AIX and DB2 UDB" on page 53 the IBM TotalStorage ESS Specialist could also have been used to verify the number of virtual paths that have been enabled for a particular ESS array.

As described in “IBM TotalStorage ESS Expert” on page 70, the ESS Expert tool can also be used to provide information on the performance of the underlying ESS subsystem. This would perhaps have been necessary in this scenario if the performance of the various vpaths (once summarized) had not been roughly the same.

As described in “DB2 UDB performance and diagnostics tools” on page 86, there are multiple views of performance that DB2 UDB’s tools provide. In this scenario, we could have used the snapshot monitor to help us understand our I/O activity and bufferpool usage.

It is worthwhile to remember that many tools will provide log files and error message files that can be reviewed. One such file is the db2diag.log file. If DB2 UDB is having trouble with containers that have been defined for its use, quite often the information about which containers are involved will be in the db2diag.log file. This file can be found in the `~home/<db2instance_owner>/sqllib/db2dump` directory.

7

Database backup and recovery

In this chapter we look at how to backup and recover your database. In particular we examine how you can use the Copy Services features of the ESS with the latest features of DB2 to provide non-disruptive backups.

7.1 Use of ESS Copy Services for fast backups

The need for continuous availability requires innovative techniques for speedy backup and restore of large databases. These techniques can take advantage of features of ESS working together with DB2 to allow the near-instantaneous creation of copies of data, without compromising either data integrity or performance of online operations. ESS Copy Services can be used to create a FlashCopy or local snapshots (copies of data at a local site) and remote snapshots (data residing at a remote site or separate ESS).

7.2 Using FlashCopy to create local copies

The ESS can create a local snapshot copy using the FlashCopy function. An authorized user can request that ESS create a flashcopy of a source logical disk onto a target logical disk within the same ESS disk system. The user can initiate the flashcopy process directly through the ESS Specialist Web-based interface, or automate the process through invocation of a command-line interface (CLI) imbedded in application scripts. For further details of using ESS Copy Services please see *Implementing ESS Copy services on Unix and WindowsNT/2000,* SG24-5757. The user can also use the ESS Specialist to create tasks that spawn several flashcopies at once. ESS rapidly makes the copies available to the host application, not waiting for data to be physically moved to the target, but simply after performing a small amount of internal housekeeping (constructing some bit maps and destaging modified data in cache to the source disks.) This entire process may take a few seconds or minutes, depending on a number of factors:

- Amount of data for the source disks in non-volatile storage
- Amount of disk traffic
- Cache size
- Total amount of flashcopied data
- Number of logical disks

Keep the following thoughts in mind:

- ESS makes the flashcopy available without requiring physical movement of data from the source, hence the rapid operation. In fact, in many cases ESS may not need to create a physical copy at all.
- The entire process does not require moving data through a host server, and so you can think of this as a serverless copy. This saves processor cycles, system memory, and data paths.

Database literature often refers to the technique used by FlashCopy as "copy-on-write."

7.2.1 NOCOPY and COPY options

The storage administrator can request deferring physical copies of data by using the Nocopy option, potentially minimizing the work that the subsystem must perform.

The Nocopy option makes the copied data available immediately after the housekeeping operations mentioned above. Although the target logical disks now contain a "virtual" copy of the source data, ESS may not need to move any data to the target physical disks. In this case, the target physical disks (or the portions of storage of the underlying RAID arrays) merely contain reserved storage, just in case the data needs to be physically copied. The ESS uses the target storage only after a host write operation causes logical source and target data to differ. Until that time, the logical source and target data are identical, and the ESS uses the source physical disks for application read requests for either copy.

When an application writes data to either source or target, ESS first stores the data in cache, and notifies the application that the write is complete. Some time later, before ESS moves modified data to disk, it also moves an appropriate portion of data from the source to target physical disks. Because the data movement is a background operation, ESS avoids delaying the host write response times.

If no write operations occur to a given item of data, then ESS services read requests for either logical copy (source or target) from a single physical image of the data on the source disks. In this way, FlashCopy postpones the work of actually performing a physical copy for as long as possible. Such work may, in fact, never need to be performed, particularly if the flashcopy is intended for temporary use (for example, as a fall-back copy of data used in batch processing). Also, the physical copy operations occur over an extended period of time, rather than en mass as in a traditional copy.

The user can also request that ESS perform a physical copy through the Copy option. In that case, processing begins by performing the same simple housekeeping used in the Nocopy case, making both copies available to the host in minimal time. The ESS then performs a physical copy in background mode, while maintaining the availability of host access to either copy. The copy operation operates in the background at lower priority than host requests to minimize effect on host workloads.

If you choose to use the “nocopy” method, the association between the source and target logical disks remains established until it is manually broken. As long as this association remains, all changes are recorded, and the target is used to store only data that has been changed in the source. This approach minimizes local I/O, but it also means that your source data will be accessed not only by its normal applications, but also by any applications referencing the FlashCopy.

Under either scenario, after the FlashCopy had been initiated you could mount the copy on a second host system and spool it to tape with assurance of data integrity. All the while, the source would remain online and accessible.

7.2.2 Considerations for using FlashCopy

Remember that a FlashCopy is a byte-by-byte copy of the entire logical disk. This means that disk identification information is copied verbatim; and the AIX operating system will complain if you attempt to access what appears to be a disk that is in two places at once. To get around this, you must either attach the copy to another system; or you must use the AIX **`renamevg`** command to change the disk identification information on the copy.

Also, take care creating flashcopy tasks. Copy Services provides great flexibility in establishing source and target logical disks, and will overwrite any target logical disk you select; so you’ll want to establish mechanisms to make sure you don’t select a target that contains data you want to preserve.

If you intend to maintain primary and secondary copies of every logical disk within an ESS logical subsystem, then a simple mechanism is to use the even numbered logical disks as sources, and the odd numbered logical disks as targets. However, if you intend to serialize your backups, using the same target for several sources, you’ll have to come up with some other technique. Just try to be consistent, and communicate well.

7.3 Creating remote copies with FlashCopy and PPRC

The following technique can be used to create a remote snapshot of a database. This process is sometimes called "split-mirror backup", but in fact uses a combination of splitting remote mirrored copies of PPRC in conjunction with the copy-on-write capability of FlashCopy.

The ESS can maintain a mirrored copy of a primary logical disk in a remote ESS disk system (the remote copy is the secondary disk) through the Peer-to-peer Remote Copy (PPRC) function. PPRC provides a synchronous write protocol, which means that while the primary and secondary are in "duplex mode", an application is not notified of completion of a write request until both copies have been accepted into cache memory of both the local and remote systems. As far as the application is concerned, the primary and secondary disk always remain perfectly synchronized.

However, this synchronization can be suspended (the logical disk pair are changed from duplex mode to suspend mode), meaning that changes to the primary disk are temporarily not propagated to the secondary. While in suspend mode, the ESS keeps track of what data has changed, and propagates those changes to the secondary logical disk some time later when duplex mode is "resumed". While the disk pair is in suspend mode, the backup procedure can initiate a flashcopy of the secondary logical disk. (Actually, the disk pair need not be suspended to do this, but there are performance reasons that make this a good idea.) When the flashcopy reaches logical completion (which as described earlier takes a few seconds or minutes), the procedure can then resume the duplex mode, which re synchronizes the primary and secondary. You can then use the target of the FlashCopy at the remote site as your snapshot copy. As with FlashCopy, your backup process can control these functions manually through the ESS Specialist or more likely through scripts that use the CLI.

7.4 DB2 backup options with Copy Services

When you create a flashcopy (locally or remotely), ESS creates a logical copy of a logical disk (or set of disks) over a window of time without freezing or inhibiting application changes to those disks, and therefore requires proper synchronization with the database system. DB2 provides capabilities to ensure this synchronization.

7.4.1 Performing cold or hot backups

An offline backup requires shutting down the database server. However, as mentioned previously, with use of ESS Copy Services, the amount of time that the database server must be shut down is relatively brief, being the time required to quiesce the database, flush all pending writes and then perform the FlashCopy of all the relevant volumes on the ESS.

Beginning with version 7.1, fix pack 2 of DB2, two new commands are available that can be used to facilitate online backups of DB2. When the following command is issued, database writes are stopped, but the database remains online and available for reads:

```
db2 set write suspend for database
```

Once the source database has been quiesced, a FlashCopy backup can be started. The source database can then be fully enabled again by issuing the command:

```
db2 set write resume for database
```

Any changes directed toward the data during the **write suspend** are then applied.

This gives you several permutations to choose from when implementing a backup strategy:

- Hot or cold backups of DB2
- Full (“background”) or “no copy” versions of a local FlashCopy
- Use of PPRC with or without a remote FlashCopy
- Full or “nocopy” versions of a remote FlashCopy

We won’t describe every permutation, but we will discuss three. For clarity, we should mention that a FlashCopy of your database should include both data and log files.

Beginning with version 7.2 of DB2, the ability to backup a database from a FlashCopy has been added. This will provide the ability to offload DB2 backups from the primary system to the secondary system, thus not impacting the production system at all except for the amount of time required to split a copy of the database.

With DB2 V 7.2 FixPak 4, the ability to rename a FlashCopy of a database to a different database name with different mount points for the containers has been introduced. The intent of this feature is to permit users to mount a FlashCopy version of the database on the same host as the original database, thus eliminating the need for a second host computer.

7.4.2 Offline local copy

This is by far the simplest, safest, and least expensive solution.

1. Shut down the database.
2. Start a FlashCopy, using either full or “nocopy” options.
3. Bring up the database.
4. Archive the copy to tape. If the “nocopy” option has been used, break the source-target association.

Using this scenario, the downtime of your production system is essentially the amount of time it takes to stop and restart the database, the elapsed time for a FlashCopy to complete, even a large database, is usually relatively short. When the FlashCopy is complete, it can be mounted on a second host and brought up with no fanfare, ready to be used. The advantage to bouncing your database is that doing so flushes all buffers, empties the log, and quiesces the database to a consistent state. The disadvantages are that this solutions requires the entire database to be taken offline until the FlashCopy has completed.

7.4.3 Offline remote backups

This solution is not much more difficult, and is equally as safe. The high-availability and disaster-recovery benefits of PPRC are enhanced with an immediate off-site backup as well.

1. Establish a PPRC between the production database ESS and a remote ESS for all of the DB2 tablespace containers, log directory and instance home directory.
2. shut down the production database and assure that the PPRC synchronization is complete.
3. Ensure the PPRC queue is empty and suspend the PPRC. This will ensure that ALL data has been sent to the remote site.
4. Start the production database
5. On the remote ESS, start a FlashCopy, either full or “nocopy”, of the PPRC mirror.
6. Resume the PPRC, re-establishing the mirror.
7. Archive the remote copy to tape. If the “nocopy” option has been used, break the source-target association.

If desired, the FlashCopy can be brought up as a separate database; or, instead of resuming, the PPRC relationship can be permanently broken, and the split mirror can be brought up as a separate database.

This configuration can be used as a basis for a home grown high availability (HA) solution, where the application would have to have knowledge of both systems and decide when to switch over. By using PPRC to mirror the log directory and containers, the remote site will always be up to date; however, there will be a performance penalty incurred on the primary system as each write must be acknowledged on both ESSs before it will return to the caller.

7.4.4 Online local copy

This method is very similar to that of the offline local backup with the exception that there is no need to shut down the database.

1. Issue a **`db2 set write suspend for database`** for the production database.
2. Start a FlashCopy, using either full or "nocopy" options, of both the DB2 production database tablespace containers, DB2 instance home directory and DB2 production database log directory. The CLI command that you will imbed in the script is **rsExecuteTask**
3. Once the FlashCopy has completed bring the production database back online by issuing a **`db2 set write resume for database`**
4. Archive the copy to tape. If the "nocopy" option has been used, break the source-target association.

Using this scenario, the production system is never brought down. While the database is in a write suspend mode all writes will be blocked however all readers will continue to function normally. When the FlashCopy is complete, it can be mounted on a second host and brought up with no fanfare, ready to be used.

7.4.5 Online remote backup

This solution is not much more difficult, and is equally as safe. The high-availability and disaster-recovery benefits of PPRC are enhanced with an immediate off-site backup as well.

1. Establish a PPRC between the production database ESS and a remote ESS for all of the DB2 tablespace containers, log directory and instance home directory.
2. Issue a **`db2 set write suspend for database`** for the production database and ensure that the PPRC synchronization is complete.
3. Ensure the PPRC queue is empty and suspend the PPRC. This will ensure that ALL data has been sent to the remote site.
4. Bring the production database back online by issuing a **`db2 set write resume for database`**
5. On the remote ESS, start a FlashCopy, either full or "nocopy", of the PPRC mirror.
6. Resume the PPRC, re-establishing the mirror.
7. Archive the remote copy to tape. If the "nocopy" option has been used, break the source-target association.

If desired, the FlashCopy can be brought up as a separate database; or, instead of resuming, the PPRC relationship can be permanently broken, and the split mirror can be brought up as a separate database.

Note that steps 5 and 7 (above) are optional, there is no need to FlashCopy the PPRC mirror. This configuration can be used as a basis for a home grown high availability (HA) solution, where the application would have to have knowledge of both systems and decide when to switch over. By using PPRC to mirror the log directory and containers, the remote site will always be up to date; however, there will be a performance penalty incurred on the primary system as each write must be acknowledged on both ESSs before it will return to the caller.

7.4.6 DB2 backups

Using any of the previous methods, a DB2 backup can be taken on the FlashCopied version of the database. As of DB2 V7.2 FixPak 4 there is a limitation that all tablespaces, with the exception of any temp tablespace, must use DMS containers only. At the time of printing, this restriction is scheduled to be removed with DB2 V7.2 FP #5. Outlined below are the steps required above and beyond those described earlier in this chapter.

1. Attach the FlashCopied database to another host.
2. Catalog the database.
3. Issue the following commands:

   db2start

   db2inidb database_name AS STANDBY

This would initialize and place the copy of the database in a roll forward pending mode, creating a "hot standby" database. Immediately following the db2inidb call, a full DB2 database backup can be taken, only a full database backup is supported at this time. Since DB2 is unable to determine if there was any activity on the original database at the time of the split, this backup will essentially be equivalent to a DB2 online backup even though the "online" keyword is not used. This backup can later be used to restore the production database in case of a failure.

This database will remain in roll forward pending and can sequentially apply those logs taken from the primary database after the FlashCopy had been created, leaving it in "hot standby" mode. There are several methods possible to get copies of the log files to the secondary system, including:

- Use PPRC to mirror the log files.
- Use the user exit on the secondary host to retrieved log files from the primary host's archive directory.
- Use the user exit to send a copy of the log files to the secondary host.

7.4.7 Quick copies of a DB2 database

If a quick copy of a DB2 database is required to populate a test or development system, then a FlashCopy of the production system can be utilized. The steps required are as follows:

1. Attach the FlashCopied database to another host.
2. Catalog the database.
3. Issue the following commands:

   db2start

   db2inidb database_name AS SNAPSHOT

These last commands start the database and initiate a crash recovery which applies the logs, committing completed transactions to disk and undoing transactions that were in flight during the **write suspend**.

Obviously, transactions that were in process at the moment the **write suspend** was issued will not complete. They will be backed out when the database copy is started. Nevertheless, this process can be useful for creating snapshots of the database. One example would be when you want to offload intensive query/reporting activities that did not rely on up-to-the-second information.

A DB2 backup of this copied database can be restored to the production system, however ALL log records generated after the FlashCopy completed are no longer usable. This will essentially provide a "version" level backup and it is not recommended that it be restored to the production system.

7.4.8 Using the FlashCopy as a backup

Once the FlashCopy has been taken, the copy itself can be used to restore the production system in the case of a failure where no DB2 backups are available. This can be done by following these steps:

1. Stop the DB2 production database, preferable by shutting down the entire instance, for example, db2stop
2. Mount the FlashCopy over top of the production DB2 database tablespace containers and database directory. **DO NOT** mount the DB2 log directory FlashCopy.
3. Issue the following

```
db2inidb database_name AS MIRROR
```

This will bring the FlashCopy (or split-mirror) up as in the AS STANDBY mode, at which point the surviving log files can be replayed by issuing a

```
db2 rollforward database database_name to end of logs and complete
```

Although we did not test the "As standby" and "As mirror" scenarios, we do know that a hot backup functionality with no data loss has been implemented within Tivoli Storage Manager, and we recommend that product to you (see redbook *Backing Up DB2 Using Tivoli Storage Manager*, SG24-6247).

A

Data placement tests

This appendix includes a description of the tests that were done on-site in Toronto as a part of this redbook project. These tests were executed to determine the comparative effect of various data placement options as well as to compare the effects of various database tuning parameters on performance.

While we are outlining here only the tests done directly due to the redbook internship, it is important to note that testing of ESS with DB2 UDB had already been conducted as part of a joint effort between IBM Software Group and IBM Storage Division. Benchmark work has already been completed and documented in the following whitepapers (see “DB2 UDB for AIX Performance of Enterprise Storage Server” and “DB2 UDB EEE Scalability: RS/6000 S80 & Enterprise Storage Servers”), which can be found at:
http://www-4.ibm.com/software/data/pubs/papers/index.html#udbaix:

We recognize that while we have attempted to answer some of the questions raised by this technology, there are and will continue to be additional questions to answer. For example, although we originally intended to run both SMS and DMS tests, we were unable (given our timeframe) to complete both SMS and DMS tests.

Performance — the reason why these tests were done

Since the introduction of the Enterprise Storage Subsystem many questions have been asked regarding how best to place the data to encourage good performance with DB2. Database administrators are used to working with a number of configurable parameters to tune the database's interaction with the disk I/O subsystem. However, the theory presented from the ESS point of view was that many of these parameters would not need to be tuned in order to achieve the throughput that the ESS subsystem is capable of. This is mainly due to features built into the ESS, such as write-cache, non-volatile storage, and the ability to recognize sequential patterns of disk access.

The following tests each attempted to answer specific questions. We realize that these tests represent a 'point-in-time' of both the capabilities of the ESS as well as those of DB2 UDB. For each test we will outline the question we were attempting to answer, a description of the test design, and then an overview of what the test results showed us. But first we will describe the basic hardware configuration that was used for all of these tests.

Base configuration

Each of the tests were run on an IBM RS/6000 S80 in Toronto. And utilized both the EE and EEE packages of DB2 UDB V7.2. The system is used for a variety of testing purposes and is shared. Each test was setup in shared mode but run when we had exclusive access to the system. This ensured that measurements taken included only our specific workload.

The S80 had 24 CPUs and 48 Gb of memory.

The ESS configuration was fully loaded with 48 ranks of 18 Gb drives.

The general layout of the ESS was that all arrays off of the first disk adapter in each cluster (we are referring to these as disk adapter 1 and 2) were used for data staging areas. Arrays off of the remaining disk adapters were used for Data, Index and Temp space. (We are referring to these as disk adapters 3 through 8).

In general, the arrays off of disk adapters 3 through 8 were divided into four ESS logical volumes. For most of the tests we addressed a single ESS logical disk per array, and used AIX logical volume manager to divide that ESS logical disk up into separate logical areas. In some of our tests we addressed two ESS logical disks per array. The remaining ESS logical disks were used for other performance testing projects.

Virtual path configuration

Here are the configurations of the multi-path, eight-path and three-path setups.

Multi-path setup

At the beginning of the testing, the connections from the ESS to the S80 consisted of eight SCSI connections. Multiple paths were enabled for selected logical disks only. Some of the logical disks had a single path enabled, some had four paths enabled, and some had eight paths enabled.

Eight-path setup

Once a baseline of testing was done using the multi-path setup, we decided to enable eight paths for all of the arrays which we would be using. This would ensure that (at least from an ESS / AIX perspective) we should be able to drive the system evenly across all of the resources we had available.

An important item to remember is that when the virtual pathing changes, the hdisks / vpaths are also changed. In this case, most of the hdisks in the eight-path setup referred to different underlying ESS logical disks than they had previously.

Three-path setup

During the course of the redbook internship, the system was moved to a new location. This opportunity was taken to upgrade from SCSI connections to Fibre. Three Fibre connections were made available, and all ESS logical disks were enabled with three paths.

While Fibre connections are faster than the SCSI connections were, they are not twice as fast. Since we moved from eight paths down to three, we expect to see a slight reduction in throughput capabilities of the ESS for those workloads that are pushing the limits of system throughput.

Again, all of the logical names were re-assigned. So it is important to obtain the base configuration data whenever virtual pathing changes. See Chapter 5, “Mapping ESS resources to AIX and DB2 UDB” on page 53 and Chapter 6, “Diagnostics and performance monitoring” on page 69.

Large versus small logical disks

The following discussion presents a relevant question, and then offers our test design and results.

Question

On very large data warehouse applications where the number of ESS arrays (ranks) for a single application can be large, the proliferation of hdisks (one for each virtual path and ESS logical disk), can become problematic. Is it true that logically dividing up the array using ESS will provide equivalent performance to logically dividing up the array using AIX logical volume manager? If so, customers with large data warehousing needs can use larger ESS logical disk sizes when setting up their systems without paying a performance penalty for doing so.

And a secondary question, is there a performance difference between logical disks on the same array? That is, if there is a loved application, do we need to create its logical disks in a specific array location?

Test design

This test design shows the large logical disk setup, small logical disk setup, and measurements.

Large logical disk setup

Here are the large logical disk setups for Shark, AIX, and DB2.

- **Shark setup**

  Use two arrays off of the same disk adapter. Set up one 122.7 GB ESS logical disk per array.

- **AIX setup**

  Define five logical volumes per array @12 Gb.

- **DB2 setup**

  Define DMS tablespace to consist of one container from each array — two containers. Set up five tablespaces, each with a single container from each array.

Small logical disk setup

Here are the small logical disk setups for Shark, AIX, and DB2.

- **Shark setup**

  Use two arrays off of the same disk adapter. Set up one 12 Gb ESS Logical disk per array.

- **AIX setup**

  Define logical volumes of equal size to the ESS logical disk size

- **DB2 setup**

  Define DMS tablespace to consist of one container from each array-- two containers.

Figure A-1 illustrates the large and small logical disk setups.

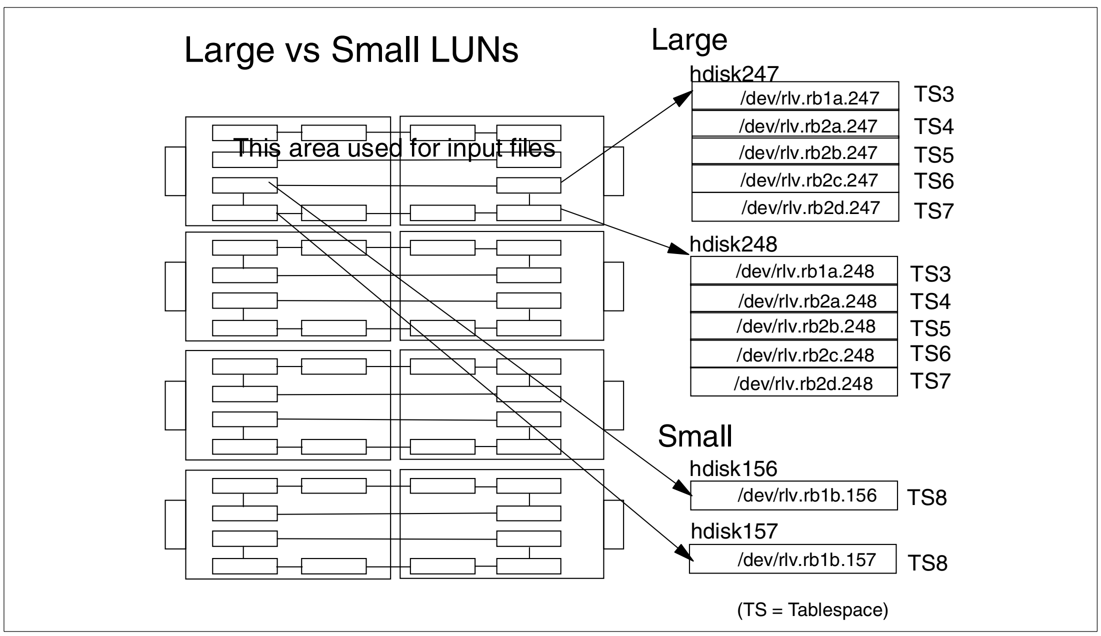


Figure A-1 Large and small logical disk layout diagram

Measurements

Measure the elapsed time it takes to load approximately 21 Gb of data into the single table tablespace. Measure the elapsed time it takes to do a tablescan. Gather iostat and vmstat data for all tests. Repeat tests to ensure variances in results are minimal.

Perform tests with a single path per array, and again with four paths enabled.

Results

As you can see in Figure A-2, the elapsed times for Loads were equivalent for all tests (less than 3% variance), regardless of placement on the arrays or the method used to 'carve up' the arrays.

However, enabling four paths compared to a single path per ESS logical disk improved the elapsed time of the loads approximately 15%.

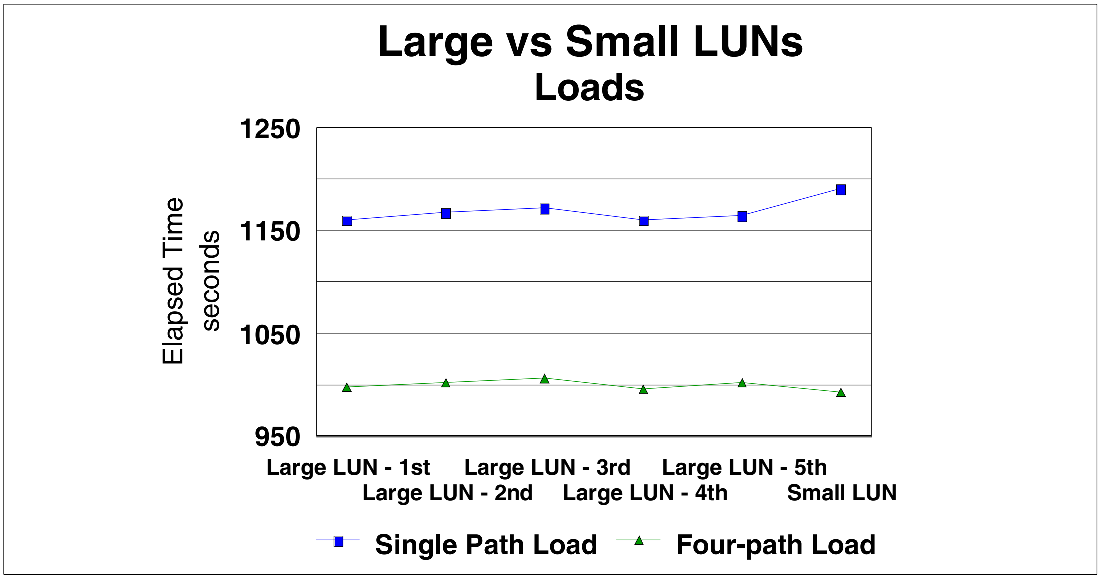


Figure A-2 Large versus small logical disks elapsed LOAD times

A tablescan was then run for each test setup using a query similar to the following (Example A-1):

Example: A-1 Tablescan query

```
select * from test.rb2a where L_EXTENDEDPRICE < 0;
```

Again, as shown in Figure A-3, the position of the logical volume within the ESS logical disk had little effect on the performance of the tablescan (less than 1% variance). While the implementation of four paths versus a single path reduced the elapsed times by 31%.

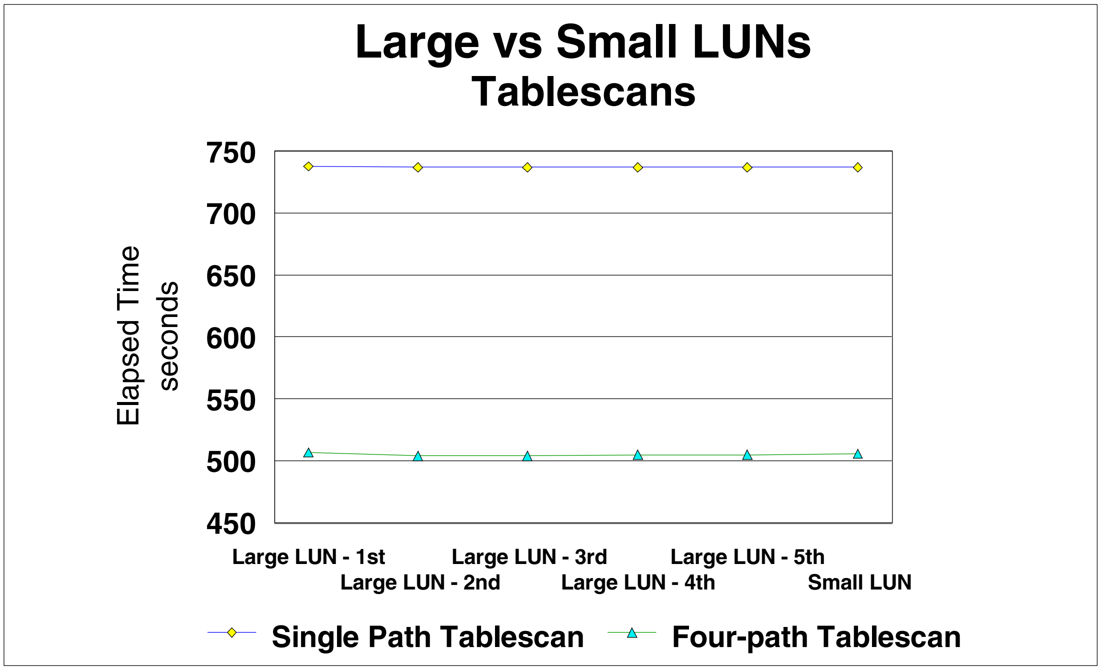


Figure A-3 Large versus small logical disks elapsed tablescan times

Vertical versus horizontal placement — 36 arrays

The following discussion presents a relevant question, and then offers our test design and results.

Question

What is the impact of 'vertical' vs. 'horizontal' layout of containers across ESS logical disks for DB2 UDB EEE? For our purposes, vertical layout means that each data partition had exclusive use of a subset of the total number of arrays (that is, arrays were not shared). A horizontal layout would mean that each data partition used all arrays (that is, arrays were shared).

If the performance is similar regardless of the association of disk to data partition, then we can be assured that there will be flexibility for growth options (for example, addition of containers) without a performance penalty due to data placement.

Test design

This test design shows the Shark, AIX and DB2 setups (vertical and horizontal) and measurements.

- **Shark setup**

  Each array off of disk adapters 3 through 8 has four ESS logical disks defined. We used one of these ESS logical disks from each array for our testing (36 logical disks). This setup did not vary for these tests.

- **AIX setup**

Each ESS logical disk was divided into multiple logical volumes, resulting in 6 raw devices per array for use in data storage, and 6 filesystems per array to use for Temporary Storage.

- **DB2 setup**

  Each test was run with data and index defined in the same DMS tablespace using an extent size of 16 pages and a prefetch size of 96 pages using a 16 KB page size.

Vertical

Seven EEE data partitions were defined. Node 0 was the catalog node, nodes 1 through 6 were data and index partitions. The data partitions each had exclusive use of six ESS logical disks (not shared). 36 containers were defined for each data partition. Figure A-4 illustrates this layout.

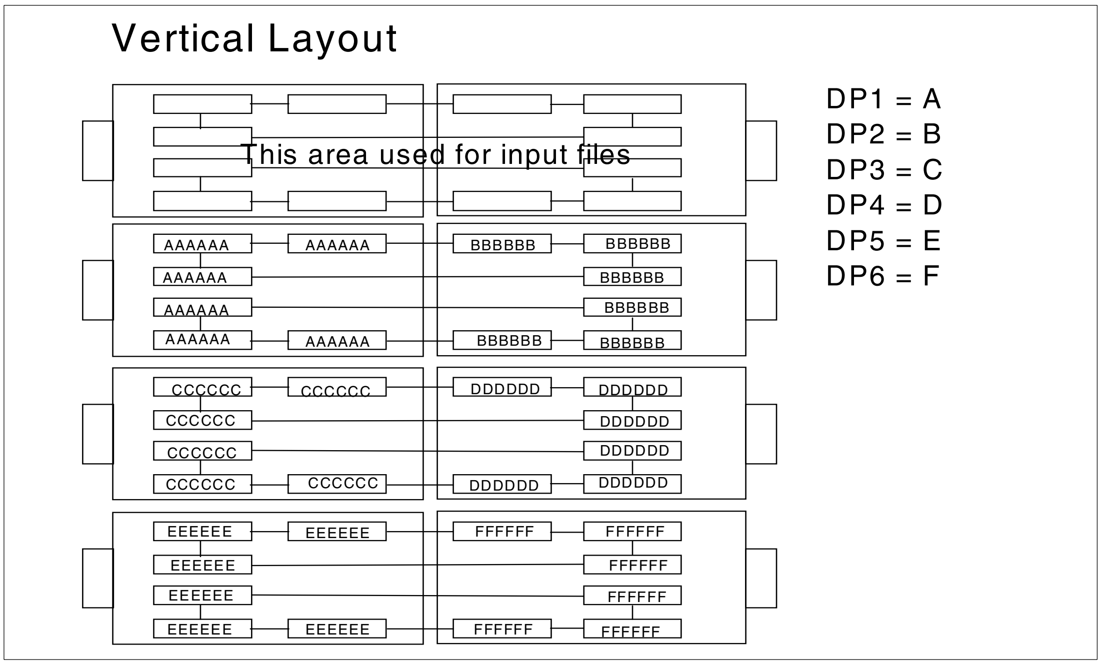


Figure A-4 Vertical layout diagram

Horizontal

Seven EEE data partitions were defined. Node 0 was the catalog node, nodes 1 through 6 were data and index partitions. The data partitions each used one container from every ESS logical disk (shared). Again, 36 containers were defined for each data partition. Figure A-5 illustrates this.

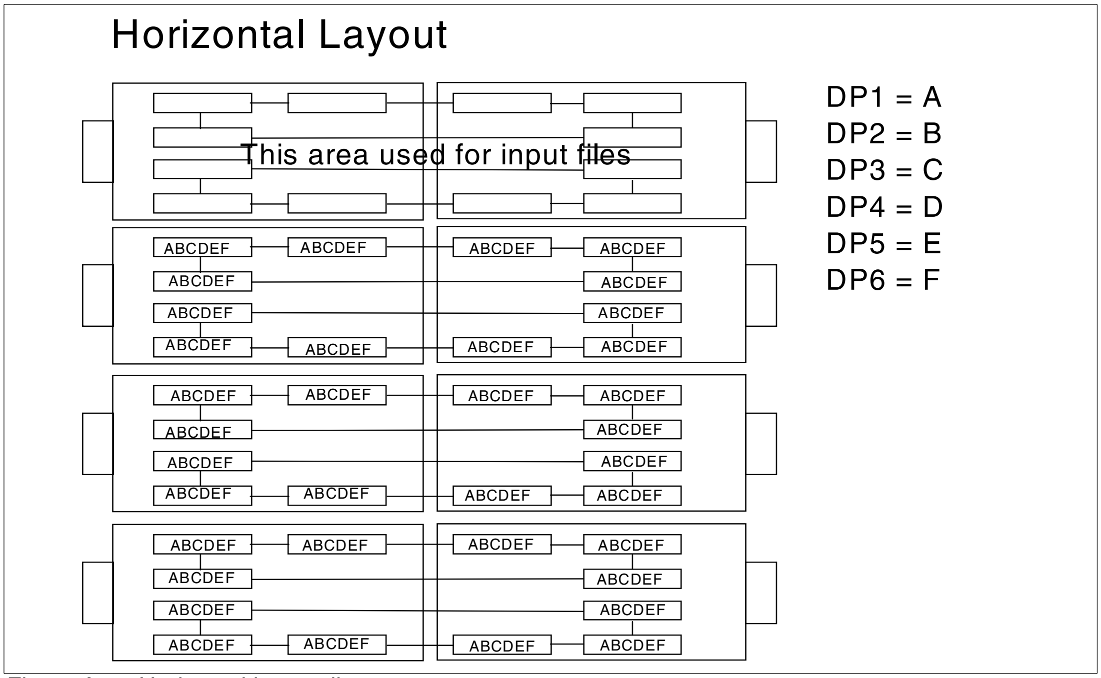


Figure A-5 Horizontal layout diagram

Measurements

Using a database of approximately 100 Gb in size, we measured the elapsed time it takes to load data into the tablespaces. We also measured the elapsed time it takes to perform a set of 22 queries. In addition elapsed time for tablescans were also measured. Iostat and vmstat data were gathered for all tests. Tests were repeated to ensure variances in results were minimal.

Results

Tests were performed with all three setups described previously (mixed-path, eight-paths, and three Fibre connections). For each setup the variance between vertical and horizontal layout was minimal. This indicates that as long as the placement of the data encompasses as many disk arms as possible and the access to the data partitions is evenly balanced, grouping disk resources by data partition does not increase performance. However, it may still be beneficial to group disk resources by data partition for other reasons, such as manageability and availability.

As you can see in Figure A-6, the load times for all tables was roughly equivalent regardless of whether or not the data was laid out ‘vertically’ or ‘horizontally’. For the mixed-path setup the variance between the two layouts was less than 2%. For the 8-path layout the variance was less than 5% and for the 3-path Fibre, the variance was less than 3%.

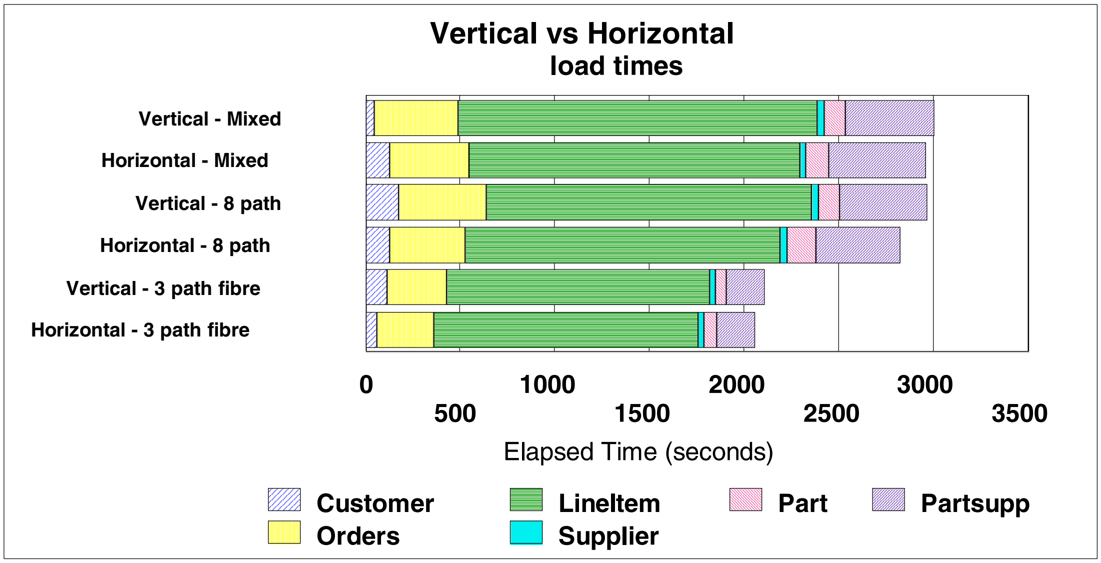


Figure A-6 Horizontal versus vertical elapsed load times

Elapsed times for queries are shown in Figure A-7. As you can see, the variance between the vertical and horizontal was greatest for the mixed-path setup-- a variance of 20%. This was due to the fact that some of the arrays only had single paths defined, which created a bottleneck for all data partitions in the horizontal layout (as opposed to affecting a sub-set of the partitions for the vertical layout). When an equal number of paths were defined for every array participating in the test, then the variance went down to 2% or lower.

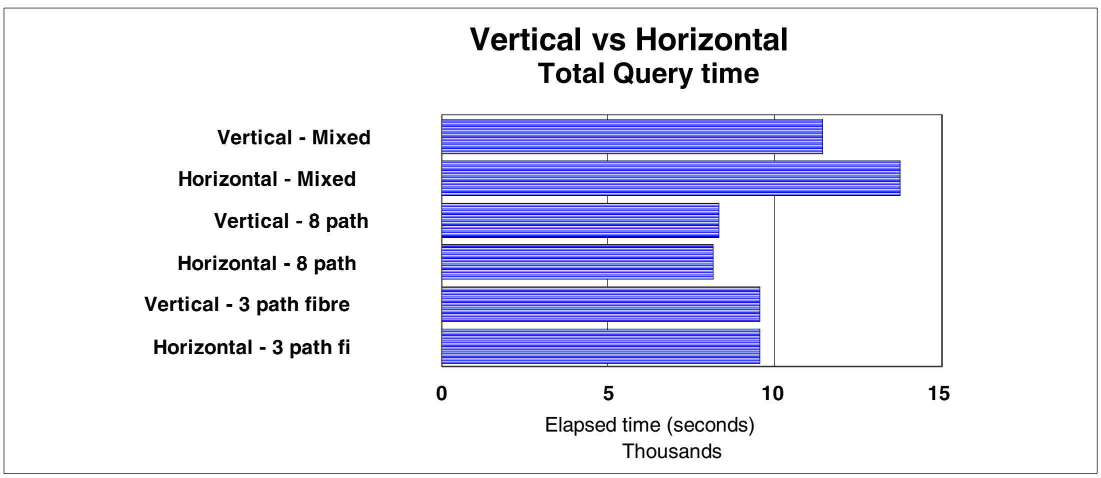


Figure A-7 Vertical versus horizontal elapsed query times

Vertical versus horizontal placement — 18 arrays

The following discussion presents a relevant question, and then offers our test design and results.

Question

Since we had run the original vertical versus horizontal testing using 36 arrays, with just a small variance in performance, we decided to take a look at using a smaller number of arrays to see if variances in layout would manifest themselves in a more restricted environment.

Test design

This test design shows the Shark, AIX and DB2 setups (vertical and horizontal) and measurements.

- **Shark setup**

  Each array off of disk adapters 3 through 8 has four ESS logical disks defined. We used two of these ESS logical disks from each array in the A loop for our testing (18 Logical disks).

- **AIX setup**

  Each ESS logical disk was divided into multiple logical volumes, resulting in 6 raw devices per array for use in data storage, and 6 filesystems per array to use for Temporary Storage.

- **DB2 setup**

  Each test was run with data and index defined in the same DMS tablespace using an extent size of 16 pages and a prefetch size of 96 pages using a 16 KB page size.

Vertical

Seven EEE data partitions were defined. Node 0 was the catalog node, nodes 1 through 6 were data and index partitions. The data partitions each had exclusive use of three ESS logical disks (not shared). The 36 containers were defined for each data partition. Figure A-8 illustrates this.

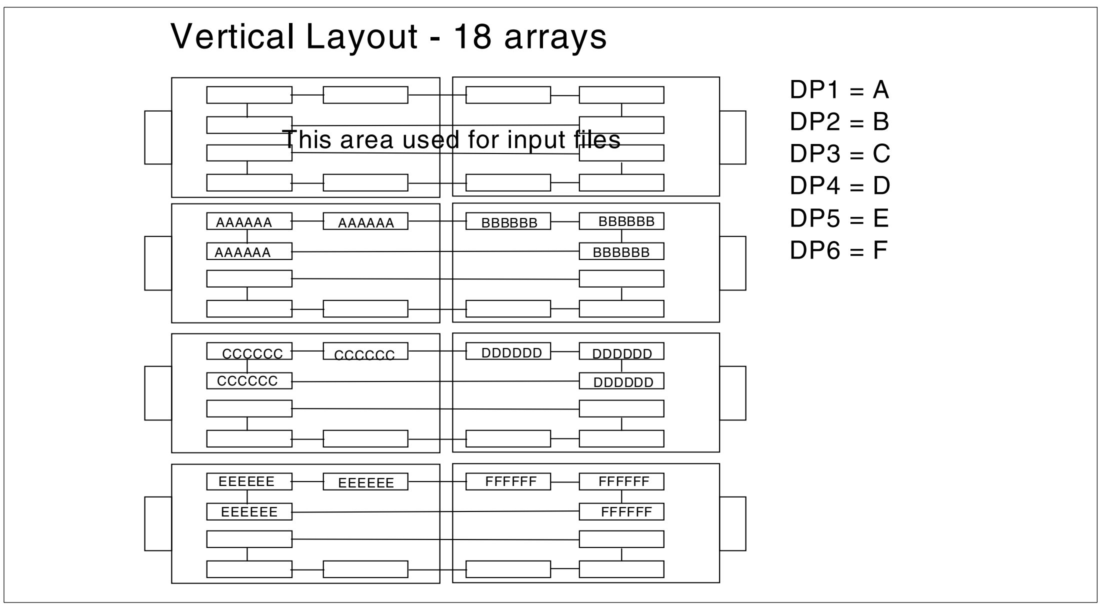


Figure A-8 Vertical layout diagram - 18 arrays

Horizontal

Seven EEE data partitions were defined. Node 0 was the catalog node, nodes 1 through 6 were data and index partitions. The data partitions each used one container from 36 ESS logical disk (shared) on 18 disk arrays. Again, 36 containers were defined for each data partition. Figure A-9 illustrates this.

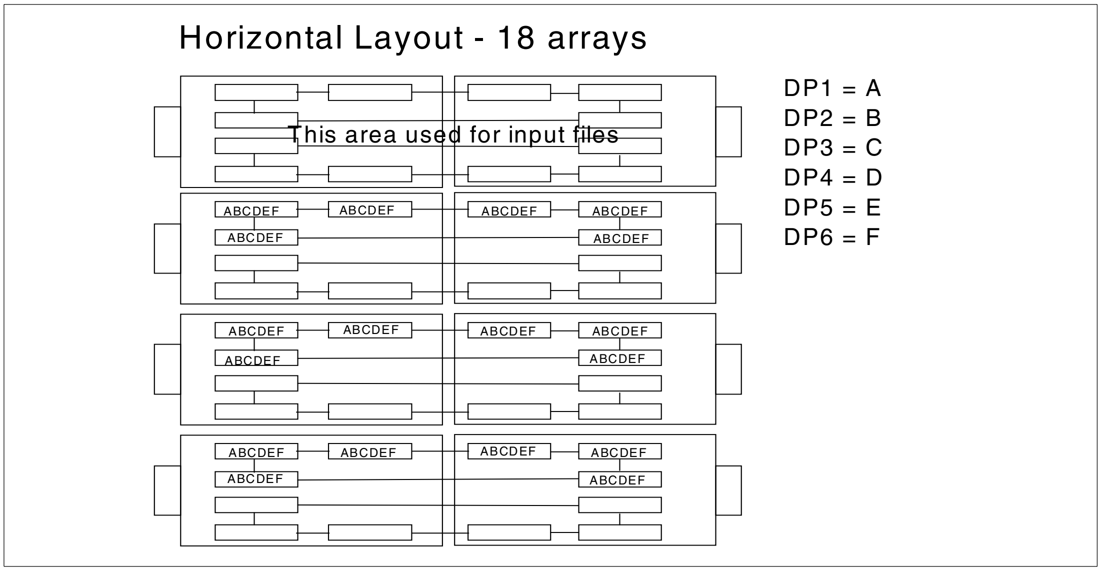


Figure A-9 Horizontal layout diagram - 18 arrays

Measurements

Using a database of approximately 100 Gb in size, we measured the elapsed time it takes to load data into the tablespaces. We also measured the elapsed time it takes to perform a set of 22 queries. In addition elapsed time for tablescans were also measured. Iostat and vmstat data were gathered for all tests. Tests were repeated to ensure variances in results were minimal.

Results

Tests were performed with three Fibre connections. The variance between vertical and horizontal layout was minimal. This indicates that even in a more restricted environment (less arrays available) as long as the access to the data partitions is evenly balanced, grouping disk resources by data partition does not increase performance. However, it may still be beneficial to group disk resources by data partition for other reasons, such as manageability and availability.

As you can see in Figure A-10, the load times for all tables was roughly equivalent regardless of whether or not the data was laid out ‘vertically’ or ‘horizontally’ with the variance less than 2% between the ‘vertical - 18 array layout’ and the ‘horizontal - 18 array layout’. We have included here for reference the elapsed load times for the horizontal test which used 36 arrays. The 36 array elapsed load times were 88-89% of the times for loading data using 18 arrays.

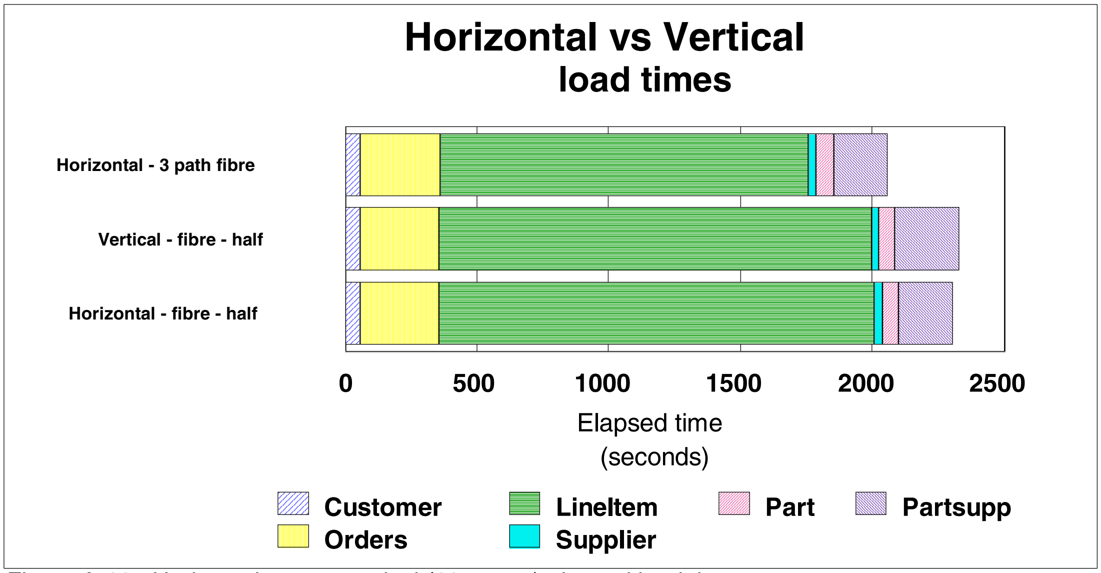


Figure A-10 Horizontal versus vertical (18 arrays) elapsed load times

Elapsed times for queries are shown in Figure A-11. Again, the layout has little impact on the total elapsed query times.

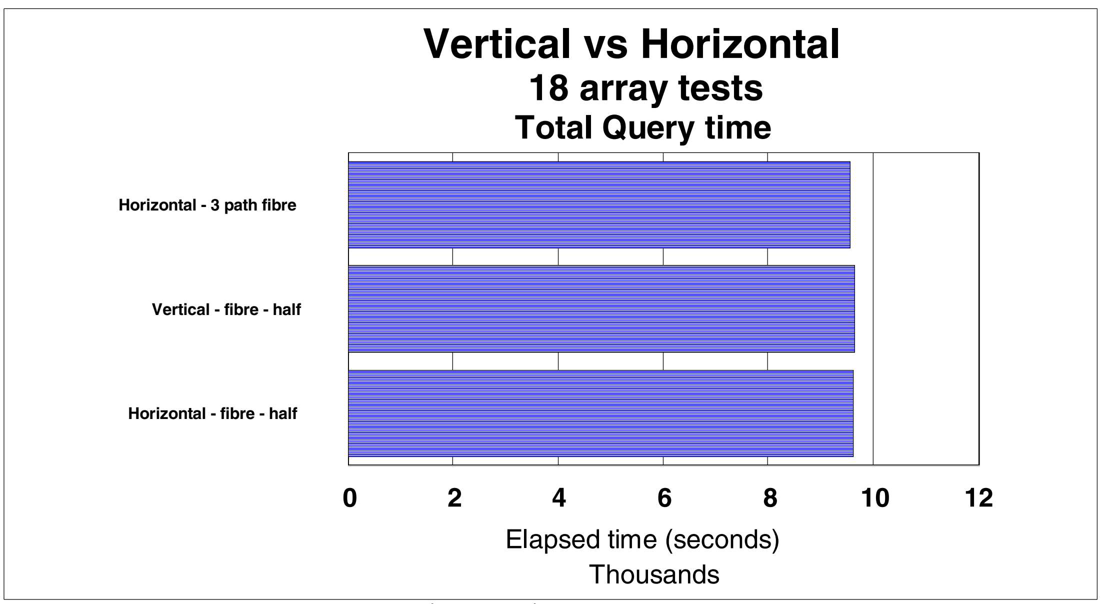


Figure A-11 Horizontal versus vertical (18 arrays) elapsed query times

Extent and prefetch size tests

The following discussion presents a relevant question, and then offers our test design and results.

Question

What is the effect of extent and prefetch size on the speed of loads and queries? Should the existing DB2 UDB recommendations remain the same for the relation of extent and prefetch sizes when using disk arrays? Prefetch is usually recommended to be the extent size times the number of containers.

Test design

This test design shows the extent and prefetch size tests and measurements.

- **Extent size tests**

  Setup using the same database definitions as per the horizontal / vertical test runs. Using a prefetch size of 144 throughout, run a series of loads / tablescans varying extent size to determine if an extent size will be optimal (or detrimental) to the performance of the ESS.

- **Prefetch size tests**

  Setup using the same database definitions as per the horizontal / vertical test runs. Run a series of tablescans keeping extent size the same as during the horizontal / vertical testing (16 extents) and varying prefetch size (64, 96 - baseline, 192, 256, and 576) to determine if a particular prefetch size will be optimal (or detrimental) to performance. Altering prefetch size does not require a re-load of the table, simply an alter tablespace statement.

Measurements

Using a database of approximately 100 Gb in size, measure the elapsed time it takes to load the data. Measure the elapsed time required to do a tablescan. Gather iostat and vmstat data for all tests. Repeat tests to ensure variances in results are minimal.

Results

Here are results for the extent and prefetch size tests.

- **Extent size tests**

  Varying extent sizes had little effect on the elapsed load times (less than 1% variance) except when the extent size was extremely low (extent size 2) which resulted in 16% longer load times. This was not surprising, as with extent size 2, DB2 UDB's I/O requests did not span a complete ESS stripe. IBM does not recommend using such a small extent size, but it was included in our testing for comparative purposes. Figure A-12 shows the load time with different extent sizes.

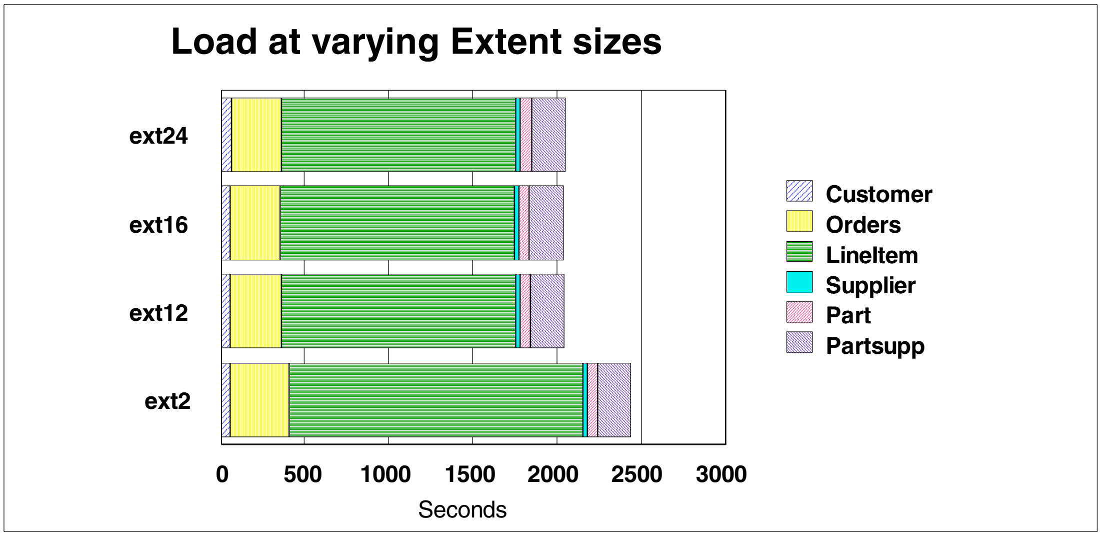


Figure A-12 Extent size testing elapsed LOAD times

It appears that as long as the combination of page size and extent and/or page size and prefetch allows for DB2 UDB to request at least one complete stripe of I/O from the array, performance will not suffer. In addition, there was no major improvement when the extent size / prefetch size combination would have been an exact multiple of the ESS stripe size.

In Figure A-13, we can see that the changes in extent size had very little effect on the query response time.

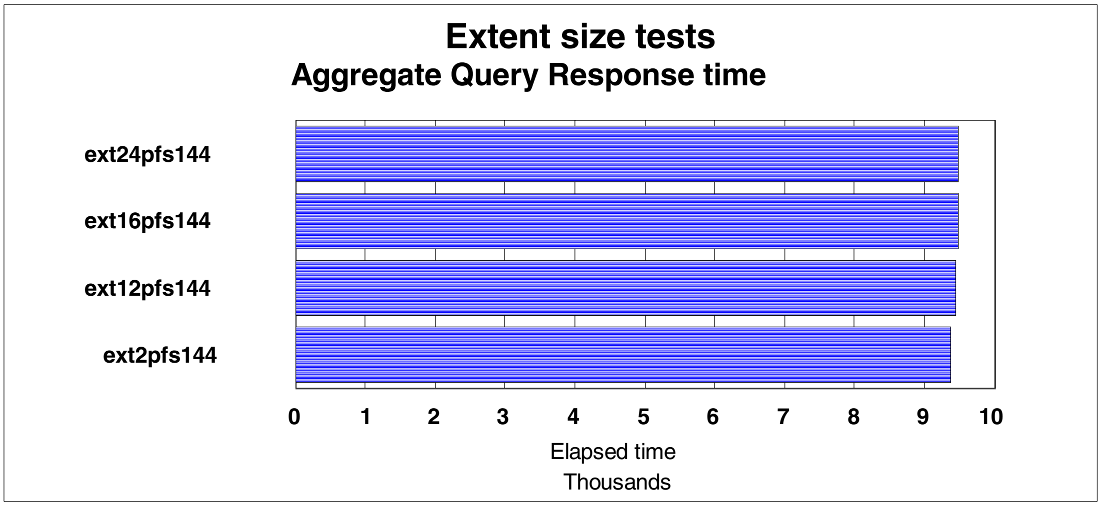


Figure A-13 Extent size tests aggregate query response time

- **Prefetch size tests**

  In Figure A-14, we see each query and how it varied based on prefetch size.

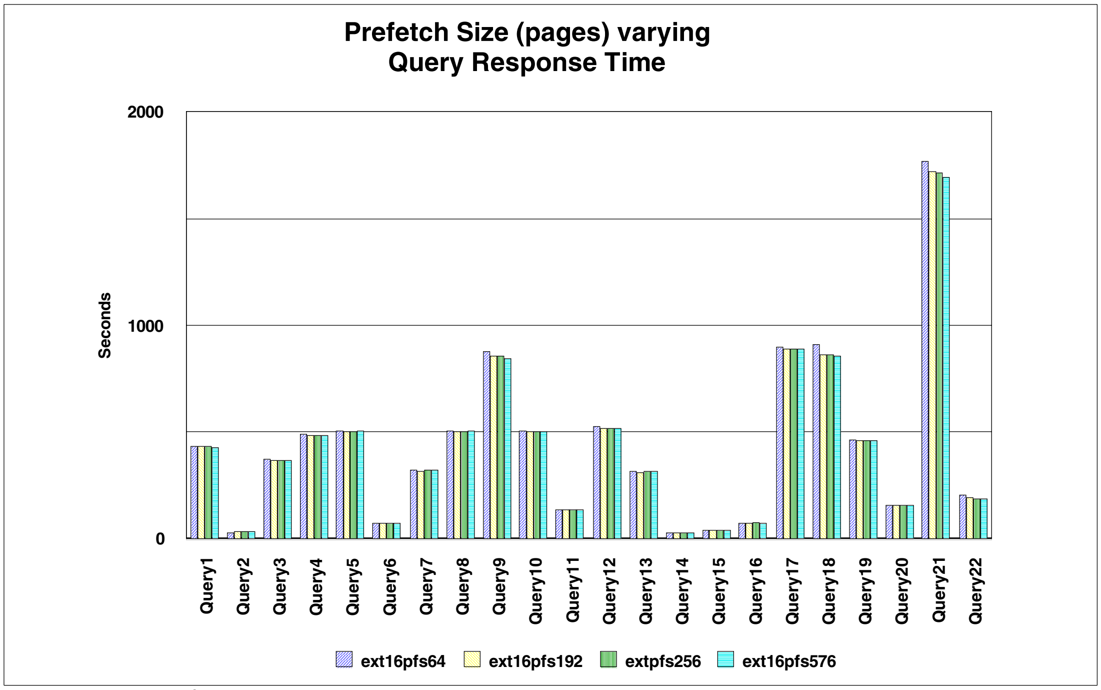


Figure A-14 Prefetch size testing - elapsed query times

Another view of this same data (Figure A-15) shows very little aggregate effect.

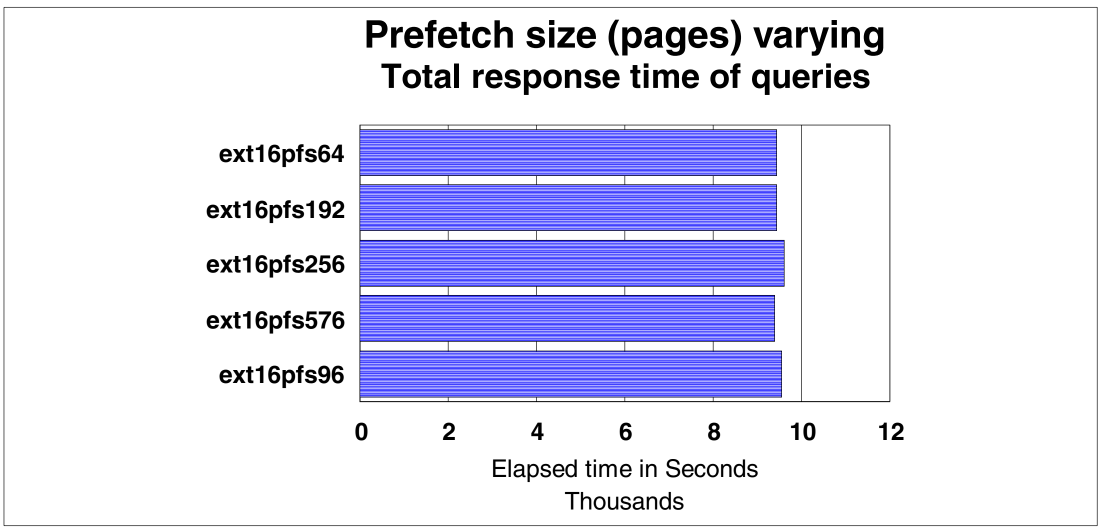


Figure A-15 Aggregate elapsed time of queries - prefetch size comparison

Refresh tests (or rolling window)

The following discussion presents a relevant question, and then offers our test design and results.

Question

All of the tests prior to these were primarily loads, tablescans and queries. We would expect that the loads and tablescans especially can take advantage of the ESS disk cache and its capabilities for interpreting sequential I/O activity. Will the layout of the data on disk (for example, horizontal versus vertical) make a difference for a write intensive workload (that is not necessarily sequential)?

Test design

Here is our test design and measurements. In addition to loading the initial tables with data as was done in the previous tests, also run refresh jobs against the tables. The refresh jobs include two main tasks: inserting data rows, and deleting data rows.

- **Inserting data**

  This includes loading an incremental number of transactions into a staging table, and then using an insert / sub-select technique to move the data into the production table. The insert job ‘chunks’ the data into smaller sections so that parallelism can be achieved.

  We loaded the original 100 Gb size database with data and then used the insert refresh job to insert an additional 1.2% of data into the tables.

  This was done using 48 chunks of data and a parallelism of 24 (this means that 24 concurrent insert subselect SQL statements were being executed at once).

- **Deleting data**

  Deletes are accomplished in a similar manner as the inserts. Rows to be deleted are ‘chunked’ into smaller subsets in order to allow multiple delete jobs to be processed in parallel.

Measurements

Using a database of approximately 100 Gb in size, we measured the elapsed time it takes to insert data from the staging table into the production tables. We also measured the elapsed time it takes to delete data from the production tables. Iostat and vmstat data were gathered for all tests. Tests were repeated to ensure variances in results are minimal.

Results

As expected, the insert / delete activity had a lower overall Kbps throughput than the tablescans did. Since tablescans are almost 100% sequential read activity, they will show the highest Kbps for the underlying arrays.

Figure A-16 shows the aggregate elapsed time it took to insert the rows from the staging table into the production table. This activity averages 62% read activity and 38% write activity. As you can see the performance of the inserts was relatively uniform regardless of the layout of the data. Variances here are slightly higher than we have seen in the previous tests with a 4% difference minimum to maximum.

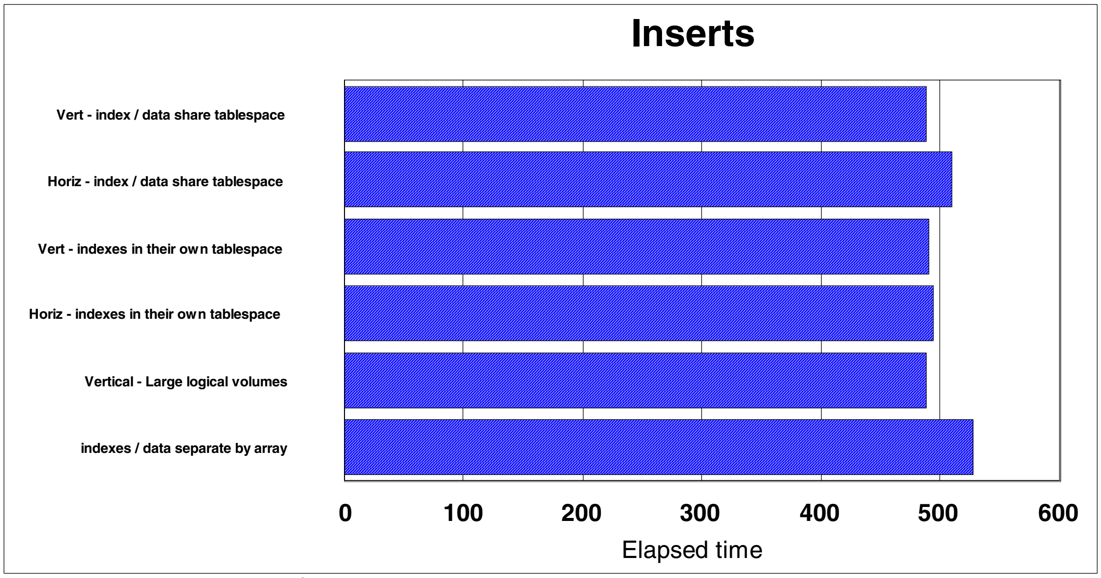


Figure A-16 Elapsed time for inserts

Deletes averaged 67% read activity and 33% write activity. The chart in Figure A-17, shows a more varied reaction to the layout of the disk for delete activity. We suppose that the reason for this is that the workload is less sequential, and therefore, less predictable to the ESS subsystem.

It is interesting to note that in general the Horizontal layouts performed better than the vertical layouts for delete activity. In addition, the single container per array layout, "Vertical - Large logical volume", performed better than it's six container per array counterpart.

The variances here are wider (34% minimum to maximum) and we would recommend that you perform more testing with your particular environment if large numbers of deletes will be a part of your normal activity.

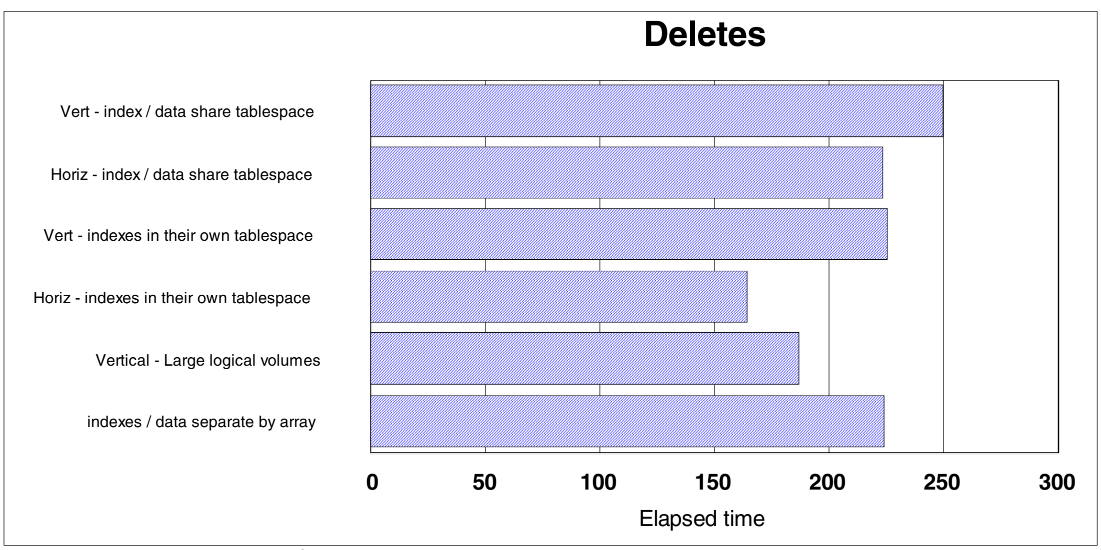


Figure A-17 Elapsed time for deletes

Six versus one container per array for DMS

The following discussion presents a relevant question, and then offers our test design and results.

Question

We had run each of the Horizontal vs. Vertical tests using 6 containers per ESS Logical disk. This resulted in 36 containers to DB2 (coming close to the actual number of underlying disks) per data partition. If we define a single container per array (encompassing the same total Gb), what will be the effect on performance?

Test design

Reallocate space on the system from the AIX perspective so that there was a single raw device per ESS logical disk. Alter the create tablespace statement to only refer to the single raw device per array. This results in 6 containers per data partition rather than 36.

Measurements

Using a database approximately 100 Gb in size, measure the elapsed time it takes to load data into the tablespaces. Measure the elapsed time it takes to perform a set of 22 queries. Measure elapsed time it takes to do a tablescan. Gather iostat and vmstat data for all tests. Repeat tests to ensure variances in results are minimal.

Results

For DMS raw, the number of containers defined to DB2 did not have a measurable effect on performance. Our expectation (and experience) indicates that this is in opposition to the behavior for SMS, although additional testing should be done.

We compared these results to the Vertical 3-path tests done earlier. As you can see in Figure A-18, the variance is insignificant, 1% or less for queries as well as for loads.

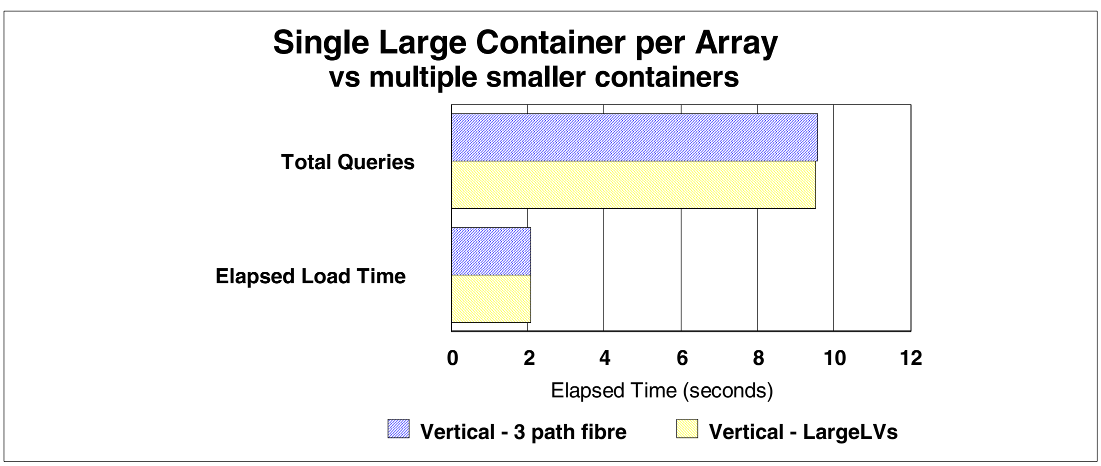


Figure A-18 Six versus one container per array elapsed times

Keeping data and index separated

The following discussion presents a relevant question, and then offers our test design and results.

Question

In the past it has been beneficial to separate data and indexes onto different physical disks. The theory behind this being that the pattern of I/O activity was different and therefore placing data onto separate physical disks from indexes allowed the disks arms to be better positioned for the next I/O activity.

With the design of the ESS, this level of separation is not necessary. However, since the question continues to arise, we decided to test this setup to see if it would indeed perform any better or worse than the tests already run.

Test design

Using the existing containers available to the tablespace, we setup one data tablespace with access to half of the containers -- all of these containers were off of the disk adapters associated with cluster 2. We then setup an index tablespace, this tablespace utilized all of the containers associated with cluster 1. This results in 18 containers per data partition rather than 36 for data, and 18 containers per data partition for indexes. However, from an overall point of view, 36 containers per array were still being accessed to accomplish the workload.

Figure A-19 illustrates this layout.

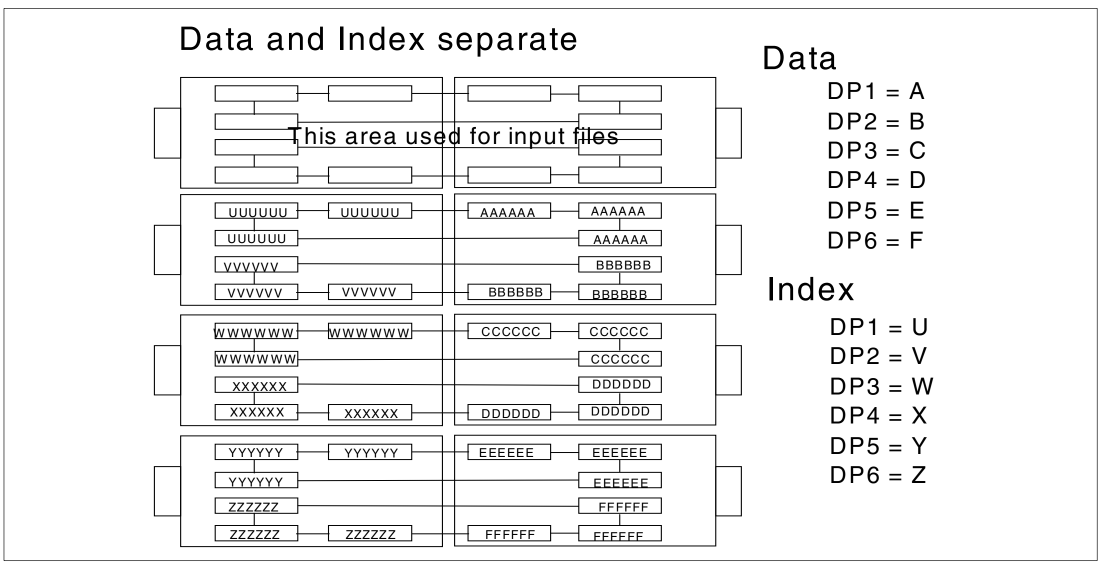


Figure A-19 Data separate from index layout diagram

Measurements

Using a database approximately 100 Gb in size, measure the elapsed time it takes to load data into the tablespaces. Measure the elapsed time it takes to do a tablescan. Gather iostat and vmstat data for all tests.

Results

In general, this setup performed worse than either a strictly vertical or a strictly horizontal layout. Figure A-16 on page 123 shows that for inserts, this setup took 4 - 8% longer to complete. However, as seen in Figure A-17 on page 123 for the deletes, this setup was in-between taking 20% longer than the best time and running 10% faster than the worst time.

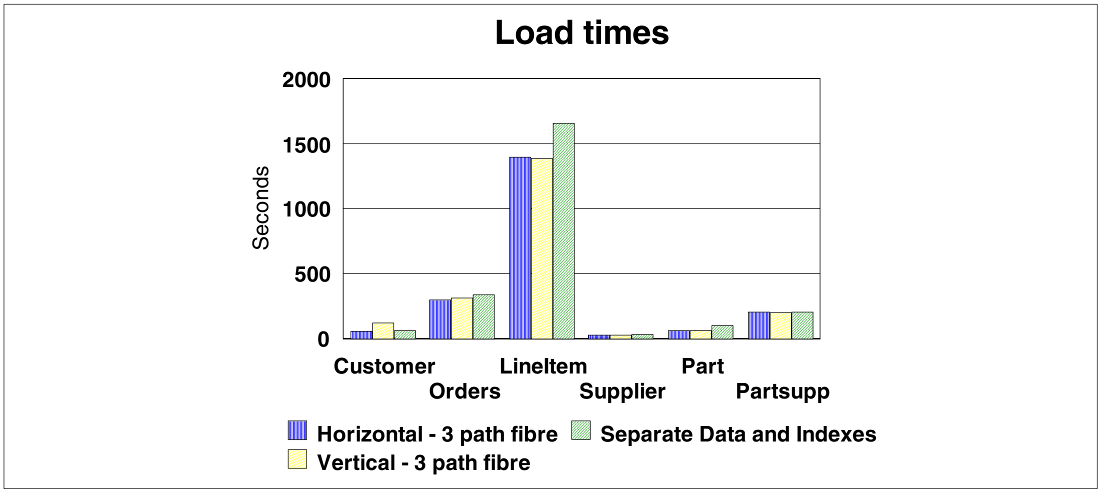


Figure A-20 Data separate from index by array - load times

As you can see in Figure A-20, for loads this setup took 12% longer than the previous tests. Tablescans using this setup took 17% longer than the previous tests.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 129.

- *Implementing the Enterprise Storage Server in Your Environment*, SG24-5420
- *IBM Enterprise Storage Server*, SG24-5465
- *IBM Enterprise Storage Server Performance Monitoring and Tuning Guide*, SG24-5656
- *Implementing ESS Copy Services on UNIX and Windows NT/2000*, SG24-5757
- *AIX 5L Differences Guide Version 5.1 Edition*, SG24-5765
- *DB2 UDB V7.1 Performance Tuning Guide*, SG24-6012
- *IBM StorWatch Expert Hands-On Usage Guide*, SG24-6102
- *Implementing Fibre Channel Attachment on the ESS*, SG24-6113
- *Backing Up DB2 Using Tivoli Storage Manager*, SG24-6247

IBM Redbooks collections

These CD-ROM are also relevant as further information sources:

- *IBM TotalStorage Redbooks Collection,* SK3T-3694-05
- *IBM Redbooks Data Management Collection,* SK2T- 8038-08
- *IBM Redbooks RS/6000 Collection,* SK2T-8043-05

Other resources

These publications are also relevant as further information sources:

- *DB2 UDB Administration Guide: Performance V7*, SC09-2945
- *DB2 UDB System Monitor Guide and Reference*, SC09-2956
- *Call Level Interface Guide and Reference V7,* SC09-2950
- *DB2 Command Reference for Common Server V2,* S20H-4645
- *DB2 SQL Reference for Common Server V2,* S20H-4665

Referenced Web sites

ESS Disk Storage Systems — Reference Material

- `http://www.storage.ibm.com/hardsoft/products/ess/refinfo.htm`

IBM Subsystem Device Driver/Data Path Optimizer on an ESS, IBM ESS Reference materials, including links to get ESSutils and related documentation

- http://SSDDOM01.storage.ibm.com/techsup/swtechsup.nsf/support/sddlink

ESS Performance White Papers

- http://www.storage.ibm.com/hardsoft/products/ess/whitepaper.htm

IBM Teraplex Integration Center: see DB2 UDB EEE scalability on RS/6000 S80 and Enterprise Storage Server

- http://www-4.ibm.com/software/data/bi/teraplex/index.htm

In-depth Kernel Overview

- http://9.101.224.11/educ/kern_adv/adv_o.fm.html

ESS Technical Support

- http://ssddom02.storage.ibm.com/disk/ess/related.html

Additional Web sites

These additional Web sites are also relevant as further information sources.

General Web sites

IBM Home Page. Use the search function to search for key words such as: ESS, DB2, AIX, SSA, SDD, PPRC, Storage, StorWatch, Copy Services, and FlashCopy.

- http://www.ibm.com

IBM Redbooks. Redbooks may be downloaded.

- http://www.redbooks.ibm.com

IBM Learning Services. For training on ESS, DB2, and AIX

- http://www-3.ibm.com/services/learning

ESS Web sites

IBM Enterprise Storage Server

- http://www.ibm.com/storage/ess

IBM TotalStorage Enterprise Storage Server — List of Supported Servers

- http://www.storage.ibm.com/hardsoft/products/ess/supserver.htm

IBM Enterprise Storage Server White Papers

- http://www.storage.ibm.com/hardsoft/products/ess/whitepaper.htm

More IBM Enterprise Storage Server White Papers

- http://www.storage.ibm.com/thought/whitepapers.html

IBM Storage Technical Support, including SDD and StorWatch

- http://www.storage.ibm.com/hardsoft/disk/techsup.htm

IBM ESS Software Technical Support

- http://ssddom01.storage.ibm.com/techsup/swtechsup.nsf/support/sddmain

IBM Hardware/Software Solutions Support Page, including SDD, Specialist, and Expert

- http://enoch.tucson.ibm.com

DB2 Web sites

IBM DB2 UDB Home Page

- http://www-4.ibm.com/software/data/db2

IBM Manuals for Data Management Products

- http://www-4.ibm.com/software/data/db2/library

IBM DB2 Product and Service Technical Library

- http://www-4.ibm.com/cgi-bin/software/db2www/library/pubs.d2w/report

IBM DB2 Online Support

- http://www-4.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/index.d2w/rep ort

Database and Data Management White Papers

- http://www-4.ibm.com/software/data/pubs/papers/index.html#udbaix

AIX Web sites

IBM AIX Home Page

- http://www-1.ibm.com/servers/aix

AIX 4.3 Documentation Library. Use the search function to search for AIX commands, such as: iostat, vmstat, filemon, and smit.

- http://www.rs6000.ibm.com/cgi-bin/ds_form

AIX 5L Documentation Library with search function

- http://publib.boulder.ibm.com/cgi-bin/ds_form?lang=en_US&viewset=AIX

AIX Mega Database

- http://rshelp.austin.ibm.com/cgi-bin/auscgi

How to get IBM Redbooks

Search for additional Redbooks or Redpieces, view, download, or order hardcopy from the Redbooks Web site:

ibm.com/redbooks

Also download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become Redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Special notices

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other

countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Glossary

This glossary contains a list of terms used within this redbook.

A

allegiance. The ESA/390 term for a relationship that is created between a device and one or more channel paths during the processing of certain condition.

allocated storage. On the ESS, this is the space that you have allocated to volumes, but not yet assigned.

application system. A system made up of one or more host systems that perform the main set of functions for an establishment. This is the system that updates the primary DASD volumes that are being copied by a copy services function.

AOM. Asynchronous operations manager.

APAR. Authorized program analysis report.

array. An arrangement of related disk drive modules that you have assigned to a group.

assigned storage. On the ESS, this is the space that you have allocated to volumes, and assigned to a port.

asynchronous operation. A type of operation in which the remote copy XRC function copies updates to the secondary volume of an XRC pair at some time after the primary volume is updated. Contrast with synchronous operation.

ATTIME. A keyword for requesting deletion or suspension at a specific target time.

availability. The degree to which a system or resource is capable of performing its normal function.

B

bay. Physical space on an ESS rack. A bay contains SCSI, ESCON or Fibre Channel interface cards and SSA device interface cards.

backup. The process of creating a copy of data to ensure against accidental loss.

C

cache. A random access electronic storage in selected storage controls used to retain frequently used data for faster access by the channel.

cache fast write. A form of fast write where the subsystem writes the data directly to cache, where it is available for later destaging.

CCA. Channel connection address.

CCW. Channel command word.

CEC. Central electronics complex.

channel. (1) A path along which signals can be sent; for example, data channel and output channel. (2) A functional unit, controlled by the processor, that handles the transfer of data between processor storage and local peripheral equipment.

channel connection address (CCA). The input/output (I/O) address that uniquely identifies an I/O device to the channel during an I/O operation.

channel interface. The circuitry in a storage control that attaches storage paths to a host channel.

channel path. The ESA/390 term for the interconnection between a channel and its associated controllers.

channel subsystem. The ESA/390 term for the part of host computer that manages I/O communication between the program and any attached controllers.

CKD. Count key data. An ES/390 architecture term for a device that specifies the format of and access mechanism for the logical data units on the device. The logical data unit is a track that can contain one or more records, each consisting of a count field, a key field (optional), and a data field (optional).

CLIST. TSO command list.

cluster. See storage cluster.

cluster processor complex (CPC). The unit within a cluster that provides the management function for the storage server. It consists of cluster processors, cluster memory, and related logic.

concurrent copy. A copy services function that produces a backup copy and allows concurrent access to data during the copy.

concurrent maintenance. The ability to service a unit while it is operational.

consistency group time. The time, expressed as a primary application system time-of-day (TOD) value, to which XRC secondary volumes have been updated. This term was previously referred to as “consistency time”.

consistent copy. A copy of data entity (for example a logical volume) that contains the contents of the entire data entity from a single instant in time.

contingent allegiance. ESA/390 term for a relationship that is created in a controller between a device and a channel path when unit-check status is accepted by the channel. The allegiance causes the controller to guarantee access; the controller does not present the busy status to the device. This enables the controller to retrieve sense data that is associated with the unit-check status, on the channel path with which the allegiance is associated.

control unit address (CUA). The high order bits of the storage control address, used to identify the storage control to the host system.

Note: The control unit address bits are set to zeros for ESCON attachments.

CUA. Control unit address.

D

daisy chain. A method of device interconnection for determining interrupt priority by connecting the interrupt sources serially.

DA. Device adapter.

DASD. Direct access storage device. See disk drive module.

data availability. The degree to which data is available when needed. For better data availability when you attach multiple hosts that share the same data storage, configure the data paths so that data transfer rates are balanced among the hosts.

data sharing. The ability of homogenous or divergent host systems to concurrently utilize information that they store on one or more storage devices. The storage facility allows configured storage to be accessible to any attached host systems, or to all. To use this capability, you need to design the host program to support data that it is sharing.

DDM. Disk drive module

data compression. A technique or algorithm that you use to encode data such that you can store the encoded result in less space than the original data. This algorithm allows you to recover the original data from the encoded result through a reverse technique or reverse algorithm.

data field. The third (optional) field of a CKD record. You determine the field length by the data length that is specified in the count field. The data field contains data that the program writes.

data record. A subsystem stores data records on a track by following the track-descriptor record. The subsystem numbers the data records consecutively, starting with 1. A track can store a maximum of 255 data records. Each data record consists of a count field, a key field (optional), and a data field (optional).

DASD-Fast Write. A function of a storage controller that allows caching of active write data without exposure of data loss by journaling of the active write data in NVS.

DASD subsystem. A DASD storage control and its attached direct access storage devices.

data in transit. The update data on application system DASD volumes that is being sent to the recovery system for writing to DASD volumes on the recovery system.

data mover. See system data mover.

dedicated storage. Storage within a storage facility that is configured such that a single host system has exclusive access to the storage.

demote. The action of removing a logical data unit from cache memory. A subsystem demotes a data unit in order to make room for other logical data units in the cache. It could also demote a data unit because the logical data unit is not valid. A subsystem must destage logical data units with active write units before they are demoted.

destage. (1) The process of reading data from cache. (2) The action of storing a logical data unit in cache memory with active write data to the storage device. As a result, the logical data unit changes from cached active write data to cached read data.

device. The ESA/390 term for a disk drive.

device address. The ESA/390 term for the field of an ESCON device-level frame that selects a specific device on a control-unit image. The one or two leftmost digits are the address of the channel to which the device is attached. The two rightmost digits represent the unit address.

device adapter. A physical sub unit of a storage controller that provides the ability to attach to one or more interfaces used to communicate with the associated storage devices.

device ID. An 8-bit identifier that uniquely identifies a physical I/O device.

device interface card. A physical sub unit of a storage cluster that provides the communication with the attached DDMs.

device number. ESA/390 term for a four-hexadecimal-character identifier, for example 13A0, that you associate with a device to facilitate communication between the program and the host operator. The device number that you associate with a subchannel.

device sparing. Refers to when a subsystem automatically copies data from a failing DDM to a spare DDM. The subsystem maintains data access during the process.

Device Support Facilities program (ICKDSF). A program used to initialize DASD at installation and perform media maintenance.

DFDSS. Data Facility Data Set Services.

DFSMSdss. A functional component of DFSMS/MVS used to copy, dump, move, and restore data sets and volumes.

director. See storage director and ESCON Director.

disaster recovery. Recovery after a disaster, such as a fire, that destroys or otherwise disables a system. Disaster recovery techniques typically involve restoring data to a second (recovery) system, then using the recovery system in place of the destroyed or disabled application system. See also recovery, backup, and recovery system.

disk drive module. The primary nonvolatile storage medium that you use for any host data that is stored within a subsystem. Number and type of storage devices within a storage facility may vary.

drawer. A unit that contains multiple DDMs, and provides power, cooling, and related interconnection logic to make the DDMs accessible to attached host systems.

DRAIN. A keyword for requesting deletion or suspension when all existing record updates from the storage control cache have been cleared.

drawer. A unit that contains multiple DDMs, and provides power, cooling, and related interconnection logic to make the DDMs accessible to attached host systems.

dump. A capture of valuable storage information at the time of an error.

dual copy. A high availability function made possible by the nonvolatile storage in cached IBM storage controls. Dual copy maintains two functionally identical copies of designated DASD volumes in the logical storage subsystem, and automatically updates both copies every time a write operation is issued to the dual copy logical volume.

duplex pair. A volume comprised of two physical devices within the same or different storage subsystems that are defined as a pair by a dual copy, PPRC, or XRC operation, and are in neither suspended nor pending state. The operation records the same data onto each volume.

E

ECSA. Extended common service area.

EMIF. ESCON Multiple Image Facility. An ESA/390 function that allows LPARs to share an ESCON channel path by providing each LPAR with its own channel-subsystem image.

environmental data. Data that the storage control must report to the host; the data can be service information message (SIM) sense data, logging mode sense data, an error condition that prevents completion of an asynchronous operation, or a statistical counter overflow. The storage control reports the appropriate condition as unit check status to the host during a channel initiated selection. Sense byte 2, bit 3 (environmental data present) is set to 1.

Environmental Record Editing and Printing (EREP) program. The program that formats and prepares reports from the data contained in the error recording data set (ERDS).

EREP. Environmental Record Editing and Printing

Program.

ERP. Error recovery procedure.

ESCD. ESCON Director.

ESCM. ESCON Manager.

ESCON. Enterprise Systems Connection Architecture. An ESA/390 computer peripheral interface. The I/O interface utilizes ESA/390 logical protocols over a serial interface that configures attached units to a communication fabric.

ESCON Director (ESCD). A device that provides connectivity capability and control for attaching any two ESCON links to each other.

extended remote copy (XRC). A hardware- and software-based remote copy service option that provides an asynchronous volume copy across storage subsystems for disaster recovery, device migration, and workload migration.

ESCON Manager (ESCM). A licensed program that provides host control and intersystem communication capability for ESCON Director connectivity operations.

F

failover. The routing of all transactions to a second controller when the first controller fails. Also see cluster.

fast write. A write operation at cache speed that does not require immediate transfer of data to a DDM. The subsystem writes the data directly to cache, to nonvolatile storage, or to both. The data is then available for destaging. Fast write reduces the time an application must wait for the I/O operation to complete.

FBA. Fixed block address. An architecture for logical devices that specifies the format of and access mechanisms for the logical data units on the device. The logical data unit is a block. All blocks on the device are the same size (fixed size); the subsystem can access them independently.

FC-AL. Fibre Channel - Arbitrated Loop. An implementation of the fibre channel standard that uses a ring topology for the communication fabric.

FCS. See fibre channel standard.

fibre channel standard. An ANSI standard for a computer peripheral interface. The I/O interface defines a protocol for communication over a serial interface that configures attached units to a communication fabric. The protocol has two layers. The IP layer defines basic interconnection protocols. The upper layer supports one or more logical protocols (for example FCP for SCSI command protocols, SBCON for ESA/390 command protocols). **fiber optic cable.** A fiber, or bundle of fibers, in a structure built to meet optic, mechanical, and environmental specifications.

fixed utility volume. A simplex volume assigned by the storage administrator to a logical storage subsystem to serve as working storage for XRC functions on that storage subsystem.

FlashCopy. A point-in-time copy services function that can quickly copy data from a source location to a target location.

floating utility volume. Any volume of a pool of simplex volumes assigned by the storage administrator to a logical storage subsystem to serve as dynamic storage for XRC functions on that storage subsystem

G

GB. Gigabyte.

gigabyte. 1 073 741 824 bytes.

group. A group consist of eight DDMs. Each DDM group is a raid array.

GTF. Generalized trace facility.

H

HA. Home address, host adapter, or high availability.

hard drive. A storage medium within a storage server used to maintain information that the storage server requires.

HDA. Head and disk assembly. The portion of an HDD associated with the medium and the read/write head.

HDD. Head and disk drive.

home address. A nine-byte field at the beginning of a track that contains information that identifies the physical track and its association with a cylinder.

host adapter. A physical sub unit of a storage controller that provides the ability to attach to one or more host I/O interfaces.

I

ICKDSF. See Device Support Facilities program.

identifier (ID). A sequence of bits or characters that identifies a program, device, storage control, or system.

IML. Initial microcode load.

initial microcode load (IML). The act of loading microcode.

I/O device. An addressable input/output unit, such as a direct access storage device, magnetic tape device, or printer.

I/O interface. An interface that you define in order to allow a host to perform read and write operations with its associated peripheral devices.

implicit allegiance. ESA/390 term for a relationship that a controller creates between a device and a channel path, when the device accepts a read or write operation. The controller guarantees access to the channel program over the set of channel paths that it associates with the allegiance.

Internet Protocol (IP). A protocol used to route data from its source to its destination in an Internet environment.

invalidate. The action of removing a logical data unit from cache memory because it cannot support continued access to the logical data unit on the device. This removal may be the result of a failure within the storage controller or a storage device that is associated with the device.

IPL. Initial program load.

ITSO. International Technical Support Organization.

J

JCL. Job control language.

Job control language (JCL). A problem-oriented language used to identify the job or describe its requirements to an operating system.

journal. A checkpoint data set that contains work to be done. For XRC, the work to be done consists of all changed records from the primary volumes. Changed records are collected and formed into a “consistency group”, and then the group of updates is applied to the secondary volumes.

K

KB. Kilobyte.

key field. The second (optional) field of a CKD record. The key length is specified in the count field. The key length determines the field length. The program writes the data in the key field. The subsystem uses this data to identify or locate a given record.

keyword. A symptom that describes one aspect of a program failure.

kilobyte (KB). 1 024 bytes.

km. Kilometer.

L

LAN. See local area network.

least recently used. The algorithm used to identify and make available the cache space that contains the least-recently used data.

licensed internal code (LIC).

(1) Microcode that IBM does not sell as part of a machine, but licenses to the customer. LIC is implemented in a part of storage that is not addressable by user programs. Some IBM products use it to implement functions as an alternative to hard-wired circuitry.

(2) LIC is implemented in a part of storage that is not addressable by user programs. Some IBM products use it to implement functions as an alternative to hard-wired circuitry.

link address. On an ESCON interface, the portion of a source, or destination address in a frame that ESCON uses to route a frame through an ESCON director. ESCON associates the link address with a specific switch port that is on the ESCON director. Equivalently, it associates the link address with the channel-subsystem, or controller-link-level functions that are attached to the switch port.

link-level facility. ESCON term for the hardware and logical functions of a controller or channel subsystem that allows communication over an ESCON write interface and an ESCON read interface.

local area network (LAN). A computer network located on a user's premises within a limited geographical area.

logical address. On an ESCON interface, the portion of a source or destination address in a frame used to select a specific channel-subsystem or control-unit image.

logical data unit. A unit of storage which is accessible on a given device.

logical device. The functions of a logical subsystem with which the host communicates when performing I/O operations to a single addressable-unit over an I/O interface. The same device may be accessible over more than one I/O interface.

logical disk drive. See logical volume.

logical subsystem. The logical functions of a storage controller that allow one or more host I/O interfaces to access a set of devices. The controller aggregates the devices according to the addressing mechanisms of the associated I/O interfaces. One or more logical subsystems exist on a storage controller. In general, the controller associates a given set of devices with only one logical subsystem.

logical unit. The SCSI term for a logical disk drive.

logical unit number. The SCSI term for the field in an identifying message that is used to select a logical unit on a given target.

logical partition (LPAR). The ESA/390 term for a set of functions that create the programming environment that is defined by the ESA/390 architecture. ESA/390 architecture uses this term when more than one LPAR is established on a processor. An LPAR is conceptually similar to a virtual machine environment except that the LPAR is a function of the processor. Also the LPAR does not depend on an operating system to create the virtual machine environment.

logical volume. The storage medium associated with a logical disk drive. A logical volume typically resides on one or more storage devices. A logical volume is referred to on an AIX platform as an hdisk, an AIX term for storage space. A host system sees a logical volume as a physical volume.

LSS. See logical subsystem.

LUN. See logical unit number.

least-recently used (LRU). A policy for a caching algorithm which chooses to remove the item from cache which has the longest elapsed time since its last access.

M

MB. Megabyte.

megabyte (MB). 1 048 576 bytes.

metadata. Internal control information used by microcode. It is stored in reserved area within disk array. The usable capacity of the array take care of the metadata.

million instructions per second (MIPS). A general measure of computing performance and, by implication, the amount of work a larger computer can do. The term is used by IBM and other computer manufacturers . For large servers or mainframes, it is also a way to measure the cost of computing: the more MIPS delivered for the money, the better the value.

MTBF. Mean time between failures. A projection of the time that an individual unit remains functional. The time is based on averaging the performance, or

projected performance, of a population of statistically independent units. The units operate under a set of conditions or assumptions.

Multiple Virtual Storage (MVS). One of a family of IBM operating systems for the System/370 or System/390 processor, such as MVS/ESA.

MVS. Multiple Virtual Storage.

N

nondisruptive. The attribute of an action or activity that does not result in the loss of any existing capability or resource, from the customer's perspective.

nonvolatile storage (NVS). Random access electronic storage with a backup battery power source, used to retain data during a power failure. Nonvolatile storage, accessible from all cached IBM storage clusters, stores data during DASD fast write, dual copy, and remote copy operations.

NVS. Nonvolatile storage.

O

open system. A system whose characteristics comply with standards made available throughout the industry, and therefore can be connected to other systems that comply with the same standards.

operating system. Software that controls the execution of programs. An operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

orphan data. Data that occurs between the last, safe backup for a recovery system and the time when the application system experiences a disaster. This data is lost when either the application system becomes available for use or when the recovery system is used in place of the application system.

P

path group. The ESA/390 term for a set of channel paths that are defined to a controller as being associated with a single LPAR. The channel paths are in a group state and are on-line to the host.

path-group identifier. The ESA/390 term for the identifier that uniquely identifies a given LPAR. The path-group identifier is used in communication between the LPAR program and a device to associate the path-group identifier with one or more channel paths. This identifier defines these paths to the control unit as being associated with the same LPAR.

partitioned data set extended (PDSE). A system-managed, page-formatted data set on direct access storage.

P/DAS. PPRC dynamic address switching.

PDSE. Partitioned data set extended.

peer-to-peer remote copy (PPRC). A hardware based remote copy option that provides a synchronous volume copy across storage subsystems for disaster recovery, device migration, and workload migration.

pending. The initial state of a defined volume pair, before it becomes a duplex pair. During this state, the contents of the primary volume are copied to the secondary volume.

pinned data. Data that is held in a cached storage control, because of a permanent error condition, until it can be destaged to DASD or until it is explicitly discarded by a host command. Pinned data exists only when using fast write, dual copy, or remote copy functions.

port. (1) An access point for data entry or exit. (2) A receptacle on a device to which a cable for another device is attached.

PPRC. Peer-to-peer remote copy.

PPRC dynamic address switching (P/DAS). A software function that provides the ability to dynamically redirect all application I/O from one PPRC volume to another PPRC volume.

predictable write. A write operation that can cache without knowledge of the existing formatting on the medium. All writes on FBA DASD devices are predictable. On CKD DASD devices, a write is predictable if it does a format write for the first record on the track.

primary device. One device of a dual copy or remote copy volume pair. All channel commands to the copy logical volume are directed to the primary device. The data on the primary device is duplicated on the secondary device. See also secondary device.

PTF. Program temporary fix.

R

RACF. Resource access control facility.

rack. A unit that houses the components of a storage subsystem, such as controllers, disk drives, and power.

RAID levels:

RAID-0

RAID-0 is a technique that stripes data evenly across all disk drives in the array. Strictly, it is not a RAID level, as no redundancy is provided.

RAID-1

RAID-1 provides fault tolerance by mirroring one drive to another drive. The mirror drive ensures access to data should a drive fail.

RAID-3

Originally, the Berkeley definitions of RAID levels specified that RAID-3 arrays stripe bytes across the disks. More recently, the RAID Advisory Board definitions allow for block striping across the drives.

RAID-5

RAID-5 offers an optimal balance between price and performance for most commercial server workloads. RAID-5 provides single-drive fault tolerance by implementing a technique called single equation single unknown. This technique says that if any single term in an equation is unknown, the equation can be solved to exactly one solution. The RAID-5 controller calculates a checksum using a logic function known as an exclusive-or (XOR) operation. The checksum is the XOR of all data elements in a row. The XOR result can be performed quickly by the RAID controller hardware and is used to solve for the unknown data element. A significant benefit of RAID-5 is the low cost of implementation, especially for configurations requiring a large number of disk drives. To achieve fault tolerance, only one additional disk is required. The checksum information is evenly distributed over all drives, and checksum update operations are evenly balanced within the array. To optimize RAID-5 performance the ESS utilizes a nonvolatile fastwrite cache and advanced destaging algorithms.

RAID-10

RAID-10 is a relatively new RAID level. It provides striping of data across several RAID-1 sub-arrays. That is the data is striped across mirrored pairs of disks.

random access. A mode of accessing data on a medium in a manner that requires the storage device to access nonconsecutive storage locations on the medium.

read hit. When data requested by the read operation is in the cache.

read miss. When data requested by the read operation is not in the cache.

recovery. The process of rebuilding data after it has been damaged or destroyed. In the case of remote copy, this involves applying data from secondary volume copies.

recovery system. A system that is used in place of a primary application system that is no longer available for use. Data from the application system must be available for use on the recovery system. This is usually accomplished through backup and recovery techniques, or through various DASD copying techniques, such as remote copy.

remote copy. A storage-based disaster recovery and workload migration function that can copy data in real time to a remote location. Two options of remote copy are available. See peer-to-peer remote copy and extended remote copy.

reserved allegiance. ESA/390 term for a relationship that is created in a controller between a device and a channel path, when a Sense Reserve command is completed by the device. The allegiance causes the control unit to guarantee access (busy status is not presented) to the device. Access is over the set of channel paths that are associated with the allegiance; access is for one or more channel programs, until the allegiance ends.

restore. Synonym for recover.

resynchronization. A track image copy from the primary volume to the secondary volume of only the tracks which have changed since the volume was last in duplex mode.

RVA. RAMAC Virtual Array Storage Subsystem.

S

SAID. System adapter identification.

SAM. Sequential access method.

SCSI. Small Computer System Interface. An ANSI standard for a logical interface to computer peripherals and for a computer peripheral interface. The interface utilizes a SCSI logical protocol over an I/O interface that configures attached targets and initiators in a multi-drop bus topology.

SCSI ID. A unique identifier assigned to a SCSI device that is used in protocols on the SCSI interface to identify or select the device. The number of data bits on the SCSI bus determines the number of available SCSI IDs. A wide interface has 16 bits, with 16 possible IDs. A SCSI device is either an initiator or a target.

Seascape architecture. A storage system architecture developed by IBM for open system servers and S/390 host systems. It provides storage solutions that integrate software, storage management, and technology for disk, tape, and optical storage.

secondary device. One of the devices in a dual copy or remote copy logical volume pair that contains a duplicate of the data on the primary device. Unlike the primary device, the secondary device may only accept a limited subset of channel commands.

sequential access. A mode of accessing data on a medium in a manner that requires the storage device to access consecutive storage locations on the medium.

server. A type of host that provides certain services to other hosts that are referred to as clients.

service information message (SIM). A message, generated by a storage subsystem, that is the result of error event collection and analysis. A SIM indicates that some service action is required.

sidefile. A storage area used to maintain copies of tracks within a concurrent copy domain. A concurrent copy operation maintains a sidefile in storage control cache and another in processor storage.

SIM. Service information message.

simplex state. A volume is in the simplex state if it is not part of a dual copy or a remote copy volume pair. Ending a volume pair returns the two devices to the simplex state. In this case, there is no longer any capability for either automatic updates of the secondary device or for logging changes, as would be the case in a suspended state.

SMF. System Management Facilities.

SMS. Storage Management Subsystem.

SRM. System resources manager.

SnapShot copy. A point-in-time copy services function that can quickly copy data from a source location to a target location.

spare. A disk drive that is used to receive data from a device that has experienced a failure that requires disruptive service. A spare can be pre-designated to allow automatic dynamic sparing. Any data on a disk drive that you use as a spare is destroyed by the dynamic sparing copy process.

SSA. Serial Storage Architecture. An IBM standard for a computer peripheral interface. The interface uses a SCSI logical protocol over a serial interface that configures attached targets and initiators in a ring topology.

SSID. Subsystem identifier.

stacked status. An ESA/390 term used when the control unit is holding for the channel; the channel responded with the stack-status control the last time the control unit attempted to present the status.

stage. The process of reading data into cache from a disk drive module.

storage cluster. A power and service region that runs channel commands and controls the storage devices. Each storage cluster contains both channel and device interfaces. Storage clusters also perform the DASD control functions.

storage control. The component in a storage subsystem that handles interaction between processor channel and storage devices, runs channel commands, and controls storage devices.

STORAGE_CONTROL_DEFAULT. A specification used by several XRC commands and messages to refer to the timeout value specified in the maintenance panel of the associated storage control.

storage device. A physical unit which provides a mechanism to store data on a given medium such that it can be subsequently retrieved. Also see disk drive module.

storage director. In an IBM storage control, a logical entity consisting of one or more physical storage paths in the same storage cluster. See also storage path.

storage facility. (1) A physical unit which consists of a storage controller integrated with one or more storage devices to provide storage capability to a host computer. (2) A storage server and its attached storage devices.

Storage Management Subsystem (SMS). A component of MVS/DFP that is used to automate and centralize the management of storage by providing the storage administrator with control over data class, storage class, management class, storage group, aggregate group and automatic class selection routine definitions.

storage server. A unit that manages attached storage devices and provides access to the storage or storage related functions for one or more attached hosts.

storage path. The hardware within the IBM storage control that transfers data between the DASD and a channel. See also storage director.

storage subsystem. A storage control and its

attached storage devices.

string. A series of connected DASD units sharing the same A-unit (or head of string).

striping. A technique that distributes data in bit, byte, multibyte, record, or block increments across multiple disk drives.

subchannel. A logical function of a channel subsystem associated with the management of a single device.

subsystem. See DASD subsystem or storage subsystem.

subsystem identifier (SSID). A user-assigned number that identifies a DASD subsystem. This number is set by the service representative at the time of installation and is included in the vital product data.

suspended state. When only one of the devices in a dual copy or remote copy volume pair is being updated

because of either a permanent error condition or an authorized user command. All writes to the remaining functional device are logged. This allows for automatic resynchronization of both volumes when the volume pair is reset to the active duplex state.

synchronization. An initial volume copy. This is a track image copy of each primary track on the volume to the secondary volume.

synchronous operation. A type of operation in which the remote copy PPRC function copies updates to the secondary volume of a PPRC pair at the same time that the primary volume is updated. Contrast with asynchronous operation.

system data mover. A system that interacts with storage controls that have attached XRC primary volumes. The system data mover copies updates made to the XRC primary volumes to a set of XRC-managed secondary volumes.

system-managed data set. A data set that has been assigned a storage class.

T

TCP/IP. Transmission Control Protocol/Internet Protocol.

TOD. Time of day.

Time Sharing Option (TSO). A System/370 operating system option that provides interactive time sharing from remote terminals.

timeout. The time in seconds that the storage control remains in a “long busy” condition before physical sessions are ended.

timestamp. The affixed value of the system time-of-day clock at a common point of reference for all write I/O operations directed to active XRC primary volumes. The UTC format is yyyy.ddd hh:mm:ss.thmiju.

track. A unit of storage on a CKD device that can be formatted to contain a number of data records. Also see home address, track-descriptor record, and data record.

track-descriptor record. A special record on a track that follows the home address. The control program uses it to maintain certain information about the track. The record has a count field with a key length of zero, a data length of 8, and a record number of 0. This record is sometimes referred to as R0.

TSO. Time Sharing Option.

U

Ultra-SCSI. An enhanced small computer system interface.

unit address. The ESA/390 term for the address associated with a device on a given controller. On ESCON interfaces, the unit address is the same as the device address. On OEMI interfaces, the unit address specifies a controller and device pair on the interface.

Universal Time, Coordinated. Replaces Greenwich Mean Time (GMT) as a global time reference. The format is yyyy.ddd hh:mm:ss.thmiju.

utility volume. A volume that is available to be used by the extended remote copy function to perform data mover I/O for a primary site storage control’s XRC-related data.

UTC. Universal Time, Coordinated.

V

vital product data (VPD). Nonvolatile data that is stored in various locations in the DASD subsystem. It includes configuration data, machine serial number, and machine features.

volume. An ESA/390 term for the information recorded on a single unit of recording medium. Indirectly, it can refer to the unit of recording medium itself. On a non-removable medium storage device, the terms may also refer, indirectly, to the storage device that you associate with the volume. When you store multiple volumes on a single storage medium transparently to the program, you may refer to the volumes as logical volumes.

vital product data (VPD). Information that uniquely defines the system, hardware, software, and microcode elements of a processing system.

VSAM. Virtual storage access method.

VTOC. Volume table of contents.

W

workload migration. The process of moving an application’s data from one set of DASD to another for the purpose of balancing performance needs, moving to new hardware, or temporarily relocating data.

write hit. A write operation where the data requested is in the cache.

write miss. A write operation where the data requested is not in the cache.

write penalty. The term that describes the classical RAID write operation performance impact.

write update. A write operation that updates a direct access volume.

X

XDF. Extended distance feature (of ESCON).

XRC. Extended remote copy.

XRC planned-outage-capable. A storage subsystem with an LIC level that supports a software bitmap but not a hardware bitmap.

Index

IBM Redbooks

IBM ESS and IBM DB2 UDB Working Together

(0.2”spine)
0.17”<->0.473”
90<->249 pages

IBM ESS and IBM DB2 UDB Working Together

Layout DB2 UDB on the ESS and maximize performance

Map system resources to database activity on the ESS

Make Copy Services part of your backup strategy

This IBM Redbook gives you information to use the IBM Enterprise Storage Subsystem with IBM DB2 Universal Database. Most of the information presented here applies to UNIX and Windows environments. Our examples are only for AIX.

First, we supply an overview of the products that we use in our environment and introduce the terminology. Next, we discuss the configuration options available for these products and the methodology that we recommend for sizing an Enterprise Storage Subsystem for use with the DB2 Universal Database.

We also discuss the challenges of mapping the system resources to the database activity in an ESS environment, and we discuss the diagnostic tools which are available and how to use them when working with performance issues. Then we discuss backup and recovery and explain some of the new features available with DB2 UDB V7.2, which enable some of the ESS capabilities. Last, we include our test examples.

This book is intended to be read by systems administrators, storage specialists, database administrators and database specialists. We assume a base knowledge of either: ESS, AIX and/or DB2 and intend that this book will help you understand your area of expertise in a wider context.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6262-00 ISBN 0738422584